VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Detekce obsazenosti parkoviště ze stereo obrazů pomocí metody grafových řezů

Parking Lot Occupancy Detection from Multiple Images via Graph Cuts

2013 **Bc. Ondřej Kroupa**

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání diplomové práce

Student: **Bc. Ondřej Kroupa**

Studijní program: N2647 Informační a komunikační technologie

Studijní obor: 2612T025 Informatika a výpočetní technika

Téma: Detekce obsazenosti parkoviště ze stereo obrazů pomocí metody grafových řezů
Parking Lot Occupancy Detection from Multiple Images via Graph Cuts

Zásady pro vypracování:

V dnešní době je možno využít techniky zpracování obrazu v mnoha oblastech dopravy. Jedním z řešených problémů je i detekce obsazenosti parkovacích ploch. Tyto informace je možno později využít pro různé automatizované parkovací systémy, popř. v inteligentních dopravních systémech. Cílem práce je vytvořit aplikaci pro detekci volných parkovacích míst. Detekci proveďte pomocí stereo rekonstrukce obrazu s využitím metody grafových řezů. Ve své práci proveďte:

1. Popište aktuální možnosti řešení detekce volných parkovacích míst, jejich výhody a nevýhody.
2. Seznamte se s metodou stereo rekonstrukce obrazů pomocí grafových řezů.
3. Naimplementujte metodu a pokuste se ji optimalizovat k použití při detekci volných parkovacích míst.
4. Otestuje výslednou aplikaci a zhodnoťte dosažené výsledky.

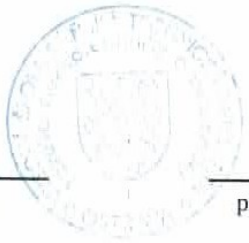
Seznam doporučené odborné literatury:

[1] Kolmogorov, Zabih and Gortler: Generalized Multi-Camera Scene Reconstruction via Graph Cuts, 2003
[2] Kolmogorov and Zabih: Multi-Camera Scene Reconstruction via Graph Cuts, 2003

Formální náležitosti a rozsah diplomové práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí diplomové práce: **Ing. Michael Holuša**

Datum zadání: 16.11.2012
Datum odevzdání: 07.05.2013



doc. Dr. Ing. Eduard Sojka
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prohlášení studenta

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

Dne: 31. července 2013

.....
podpis studenta

Poděkování

Rád bych poděkoval Ing. Michaelu Holušovi za odbornou pomoc a konzultaci při vytváření této diplomové práce.

Prohlášení zástupce spolupracující právnické nebo fyzické osoby

„Souhlasím se zveřejněním této diplomové práce dle požadavků čl. 26, odst. 9 Studijního a zkušebního řádu pro studium v magisterských programech VŠB-TU Ostrava."

Dne: 31. července 2013

.....
podpis zástupce

Abstrakt

Cílem práce je prostudování a vyzkoušení možností využití metody grafových řezů pro detekci obsazenosti parkoviště ze stereo snímků a vytvoření aplikace využívající tuto metodu. Grafové řezy jsou zde využívány k minimalizaci funkce energie stereo matchingu. Jsou zde popsány algoritmy řešící grafové řezy a kroky, které jsou potřeba k dosažení cíle. Součástí práce je také popis potřebného předzpracování snímků a jeho provedení. Programovacím jazykem této práce je C++ s použitím knihoven OpenCV 2.4.3 a gco-v3.0. Výsledný algoritmus je testován na snímcích pořízených kamerou nad parkovištěm VŠB v Ostravě-Porubě.

Klíčová slova

Grafové řezy, minimální řez grafu, maximální tok grafem, pixel labeling problém, rektifikace, detekce obsazenosti parkoviště

Abstract

The aim of this work is to study and to try possibilities of using Graph Cuts for detection of Parking Lot Occupancy from Multiple Images and creating an application using this method. Gaph Cuts are used to minimize energy function from stereo matching. There are described algorithms solving the graph cuts and the steps that are needed to achieve the goal in this work. The work also describes the necessary pre-processing images and its implementation. Programming language used here is C++ and used libraries are OpenCV 2.4.3 and gco-v3.0. The resulting algorithm is tested on images taken with the camera above the parking lot VŠB in Ostrava-Poruba.

Key words

Graph Cuts, minimal Graph Cut, maximal flow in Graph, pixel labeling problem, rectification, parking lot detection

Seznam použitých termínů

Termín	Význam termínu
label	Vzdálenost dvou odpovídajících si pixelů
disparita	Rozdíl zobrazení objektů v obrazech
labeling	Množina labelů
matching cost	Hodnota reprezentující míru shody objektů
smoothCost term	Term pro prostorovou hladkost
augmenting paths	Algoritmy zlepšující cesty

Obsah

Seznam obrázků

Seznam tabulek

1 Úvod

V této práci se budu zabývat detekcí obsazenosti volných parkovacích míst, současnými možnostmi detekce a hlavně prozkoumáním využití metody grafových řezů pro detekci obsazenosti parkovacích míst ze stereo obrazů. S tímto úkolem úzce souvisí oblast počítačového vidění – stereo matching, které se budu taktéž věnovat.

Tato oblast je dle mého názoru velice perspektivní, vzhledem k pokroku v elektronické výbavě aut včetně zavádění počítačových systémů, kdy v současné době probíhá testování a pomalé nasazování automatických parkovacích systémů, takže v případě propojení se systémem detekce volných parkovacích míst bude moct auto zaparkovat na optimální místo úplně samostatně. To je ale ještě otázkou budoucnosti, v dnešní době řidiči postačí informace o volném parkovacím místě, takže se zbaví nervů a ušetří čas při objíždění parkoviště a hledání volného parkovacího místa. Znalost obsazenosti parkoviště je stejně důležitá i pro provozovatele parkovišť, aby mohli optimalizovat jeho vytíženost, způsob parkování aut a zefektivnit tak fungování parkoviště a tím zvýšit své zisky.

Smyslem této práce je prozkoumat vhodnost a možnost využití metody grafových řezů pro detekci obsazenosti parkoviště ze stereo snímků pořízených například bezpečnostní kamerou. K dosažení tohoto cíle je potřeba nastudovat metody zpracování a úpravy obrazu a zvolit vhodnou knihovnu pro tuto práci. Dále pochopit metodu grafových řezů, algoritmy zabývající se touto problematikou a získat co nejvíce informací a poznatků z prací, které se již zabývají způsobem řešení a využití grafových řezů v počítačovém vidění. Získané poznatky poté přetavit v aplikaci pro detekci obsazenosti parkoviště.

Práci začnu prozkoumáním současné situace na poli detekce volných parkovacích míst a způsobů, jakými je řešena. V další kapitole se věnuji teorii rekonstrukce obrazu pomocí sterea a uvádím důvody, proč by mohlo být vhodné tímto způsobem řešit zaplněnost parkoviště. Dále popisuji jakými postupy a technikami budu chtít dosáhnout požadovaného cíle. V kapitole následující se věnuji praktické části a popíši vývoj a problémy které nastaly. Také uvedu výsledky rozpoznání na testovacích snímcích. Závěrečná kapitola patří zhodnocení práce, přínosu, chybám a možným vylepšením.

2 Současné způsoby detekce obsazenosti parkovacích míst

Automatická detekce volných parkovacích míst je velice užitečná a má potenciál pro případné budoucí propojení s parkovacími systémy aut. V současné době již existují různé systémy, které tento úkol řeší a pomáhají řidičům i vlastníků parkovišť. Řidiči nemusí kroužit a hledat volné místo a provozovatelé mohou dosáhnout lepších zisků díky optimálnímu využití parkovacích míst.

2.1 Monitorování pomocí čidel

Systémy založené na detekci pomocí čidel, ať už ultrazvukových, které měří změnu vzdálenosti vzhledem k zemi nebo infračervených, které detekují změny v infračerveném záření černého tělesa vydávaném objekty, mají své výhody i nevýhody. Výhodou je jednoznačně přesnost a čidlo samo o sobě má i malou spotřebu energie. Informace z čidel se mohou předávat do centrálních systémů a vyhodnocovat. Nevýhodou je, že každé parkovací místo musí mít své vlastní čidlo a adresovatelnost čidel může být omezená.

2.2 Monitorování pomocí kamer

Současné systémy detekující zaplněnost parkoviště pomocí kamer kontrolují počet aut, které na parkoviště vjely a kolik aut vyjelo, takže výsledná informace je pouze počet volných/plných míst, což může být nedostatečná informace.

3 Stereo rekonstrukce obrazu

Jedním ze způsobů rekonstrukce scény je použití stereo obrazů – dvou a více snímků téže scény, které nejsou totožné. Tento způsob je založen na vyhledání odpovídajících si bodů v daných obrazech, což není jednoduchý problém. Jeden z popisů tohoto problému se nazývá „Pixel labeling problem“ (další podobný je například „Voxel labeling problem“), který řeší hledání korespondujících pixelů pomocí grafových řezů (Graph Cuts).

3.1 Proč stereo

Hned zpočátku se nabízí otázka: „Proč stereo snímky? Nestačí znát barevné hodnoty parkovacího místa a pak je porovnávat a s aktuálními a v případě odlišnosti jde o zaplněné parkovací místo? Nebo co třeba využít detektoru hran?“. Tyto metody by pravděpodobně vedly často k chybným detekcím, byly by náchylné třeba na změnu počasí, kdy je jednou mokro jindy zase stíny přes parkovací místa nebo i opotřebování parkoviště a nebyly by tedy příliš účinné a univerzální. Naproti tomu oblast počítačového vidění *stereo matching* je pro vyhledávání a rozpoznávání objektů v obraze velice vhodná, dlouhou dobu prozkoumávána a používaná v praktických řešeních.

3.2 Stereo matching

Oblast stereo matching je jednou z nejvíce zkoumaných oblastí v počítačovém vidění, výzkumy probíhají už od roku 1962 a stále se provádí další studie, čímž je na první pohled jasné o jak důležitou a užitečnou oblast se jedná.

[1] *Stereo matching* je způsob jak zjistit 3D informace o scéně ze dvou nebo více snímků jedné scény, které jsou pořízeny ve stejný moment (nebo v co nejkratším možném časovém intervalu) z různých úhlů pohledu. Metoda je založena na schopnosti nalezení v každém z obrazů zobrazení stejného objektu. Informace o hloubce objektu je závislá na disparitě a na relativní pozici snímacích přístrojů. Při řešení detekce zaplněnosti parkoviště je možné se omezit na řešení vyhledání disparit, protože informace o přesné vzdálenosti není v tomto případě důležitá. Výpočet disparit je spojen s problémem stereo korespondence a dosud není nalezeno uspokojivé řešení, které by bylo dostatečně spolehlivé, robustní a efektivní a umožňovalo by reálné použití stereo vidění k výpočtu hloubky obrazu.

Problém stereo korespondence (matching) je obecně problém minimalizační a k jeho řešení jsou dva přístupy – lokální a globální. Lokální přístup se snaží zvlášť minimalizovat mnoho energetických funkcí reprezentujících lokální matching cost za předpokladu, že jsou nezávislé pro různé porovnávané entity a lokální matching cost závisí na podobnosti ohraničení mezi těmito entitami. Globální přístup se snaží o minimalizaci jediné energetické funkce, která zahrnuje všechny matching cost, takže vytvoří globální matching cost, který obsahuje všechny lokální matching cost a současně zahrne hodnotu, reprezentující konzistenci obrazu. Lokální metody zvlášť vyhledávají nejlepší shodu pro každý pixel obrazu bez ohledu na pixely v okolí a konzistenci. Globální metody se snaží o definování globálního modelu celé scény a korespondence mezi pixely ze stereo snímků závisí i na korespondenci jejich sousedů. Mezi globální metody se řadí i metoda grafových řezů.

Myšlenku využití grafových řezů pro řešení úloh v oblasti stereo matchingu vyslovili pánové Sébastien Roy a Ingemar J. Cox v roce 1998 [2] a na minimalizační problém energetické funkce ji přeformuloval v roce 1999 O. Veksler [3]. Poté od roku 2001 následovalo představení algoritmů používající tuto metodu od pánů V. Kolmogorova a R. Zabiha [4].

3.3 Stereo a disparita

Tato práce se celá točí kolem slova disparita, takže je by bylo dobré tento termín co nejlépe vysvětlit. Co se za ním skrývá, jaký má matematický základ a jaké má využití v oblasti počítačového vidění.

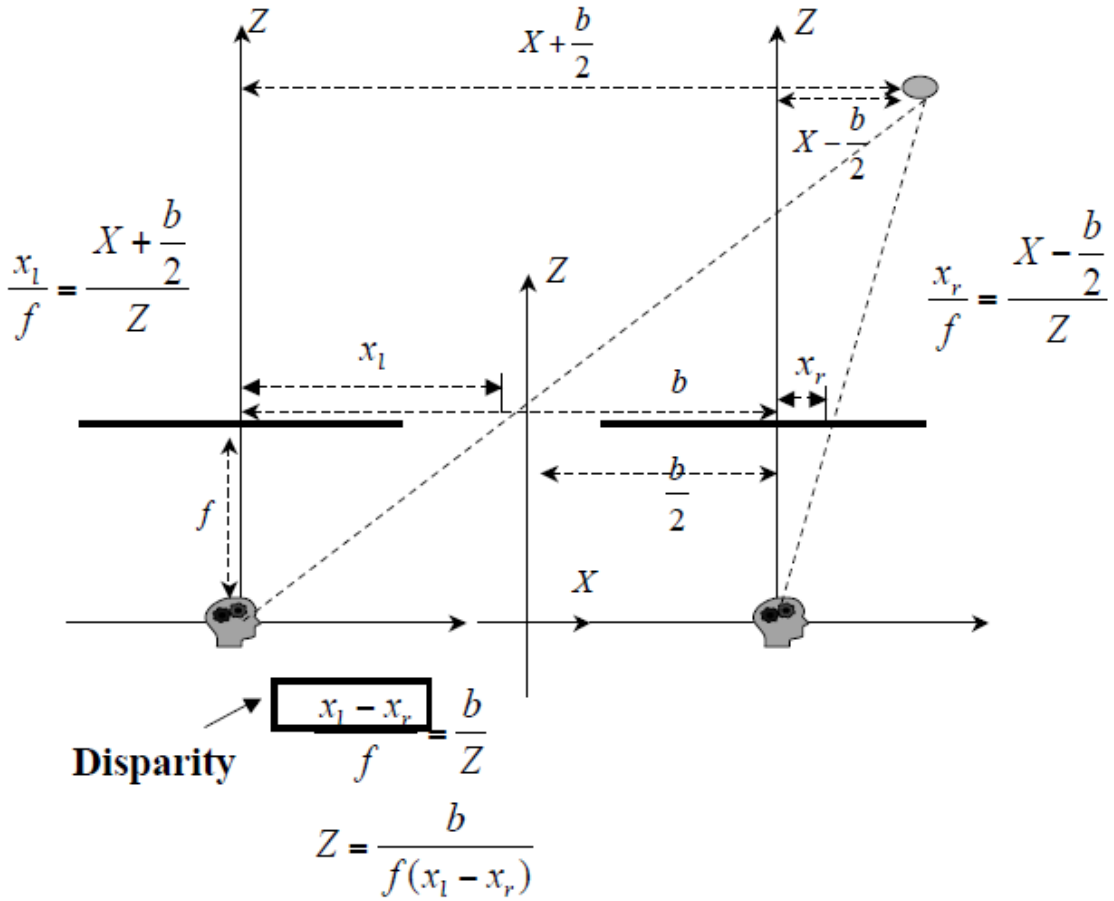
Význam disparity pro stereo matching je ten, že umožňuje zobrazit pomocí dvou stereo snímků hloubku scény. Pro počítačové vidění to znamená, že je možné ze dvou (nebo i více) 2D obrazů téže scény určit prostorové uspořádání ve scéně. Určení hloubky scény je rozhodně netriviální úloha, třebaže člověk je pouhým pohledem schopen v mnoha případech o vzájemné poloze objektů na snímku snadno rozhodnout, pro počítač to je náročná úloha. V případě tématu této práce, což je detekce zaplněnosti parkovacích míst, je právě určení hloubky obrazu klíčové – auto je vždycky nad parkovacím místem.

3.3.1 Matematický základ pro disparitu

Jako ve všem co se týká počítačového vidění má i disparita jednoznačný základ v matematice a geometrii. Na obrázku č. 1 je vidět jaký je matematický základ této myšlenky a jak se dá disparita teoreticky spočítat, pro jednoduchost a názornost je scéna pouze 2D. Vidíme tam scénu s jedním objektem, která je snímaná dvěma rektifikovanými kamerami a naším cílem je zjistit jak je ten objekt daleko, jeho hloubku. Tato hledaná vzdálenost je označena jako $\boldsymbol{Z}$. Pokud známe ohniskovou vzdálenost $\boldsymbol{f}$, vzdálenost kamer od sebe $\boldsymbol{b}$ a souřadnice (v tomto případě pouze souřadnici x) objektu na snímacích plochách obou kamer ($\boldsymbol{x_l}$ a $\boldsymbol{x_r}$), můžeme pomocí podobnosti trojúhelníků dopočítat žádanou vzdálenost objektu od promítací roviny. Výsledná rovnice pro výpočet této vzdálenosti má tvar (3.1). V této rovnici jsou známé proměnné $\boldsymbol{b}$ a $\boldsymbol{f}$, které jsou v daném případě konstantami pro jakýkoliv případný jiný objekt ve scéně, protože jde o vlastnosti kamer, takže výraz, který určuje hledanou vzdálenost je rozdíl souřadnic $\boldsymbol{x_l - x_r}$ a nazývá se disparita.

$$Z = \frac{b}{f(x_l - x_r)} \tag{3.1}$$

Teorie vypadá krásně a jednoduše, ale při aplikaci v reálném prostředí přichází netriviální problémy. Přesná vzdálenost kamer od sebe je velice těžko zjistitelná a udržitelná, protože upoutání kamer nebude nikdy dokonalé a každý pohyb nebo otřes povede ke změně vzdálenosti. Ohnisková vzdálenost je daná optickou čočkou a jejím umístěním v kameře. Výroba čočky i její následné vkládání do objektivu kamery jsou mechanické činnosti, které zanáší vady a tak žádné dvě kamery nejsou naprosto totožné. A zjištění přesných souřadnic odpovídajících si bodů v levém a pravém snímku a tím vyhledání správné disparity je kapitola sama o sobě.



Obrázek 1: Co je to disparita matematickým pohledem [10]

3.3.2 Disparita shrnutí

V této kapitole jsem popsal pojem disparita a její význam pro počítačové vidění. Protože je to pojem pro tuto práci velice důležitý a často se v ní vyskytuje, dovolím si shrnutí.

Disparita je výraz, kterým je označována vzdálenost dvou odpovídajících si bodů jednoho objektu ve stereo snímcích a reflektuje hloubku obrazu, což je důležitý faktor v oboru zpracování obrazu. Disparita je inverzní vzhledem ke vzdálenosti objektu, což znamená, že blízké objekty mají disparitu velkou a objekty vzdálené naopak malou. V počítačovém vidění je používána k určování uspořádání objektů ve scéně a jejich detekci. Její nalezení je ovšem komplikovaný problém.

Pro detekci zaplněnosti parkoviště je vhodné disparity využít, protože pokud vytvoříme mapu disparit neboli výškovou mapu scény parkoviště, bude mít prázdné parkovací místo menší výšku/větší disparitu než parkovací místo obsazené autem a mělo by tedy být možné takto obsazené místo detekovat.

3.4 Toky v sítích

Abych se dostal ke grafovým řezům, je třeba začít u toků v sítích. V síťových tocích se vyskytují dva důležitě pojmy. Jsou to minimální tok a maximální řez, které říkají, jak moc něčeho (v tomto případě energie) mohu dostat z místa A do místa B skrze síť, která tato dvě místa propojuje a která místa jsou v této síti nejužší.

[9] Síť *S* je definována jako čtveřice *S(G,s,t,c)*, kde *G=(V, E)* je orientovaný graf s množinou vrcholů *V* a množinou hran *E*, *s* (zdroj) a *t* (spotřebič) jsou speciální vrcholy náležící do množiny *V* a každá hrana *e* z množiny *E* má kapacitu *c(e)*, kde kapacita je funkce $c: E \to R^+$.

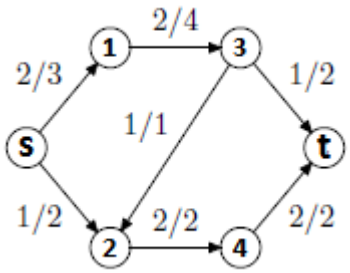
Tok *f* v síti je definován jako funkce $f: E \to R^+$, které splňuje následující podmínky:

1. $0 \leq f(e) \leq c(e), \forall e \in E$ - podmínka říká, že průtok hranou je nezáporný a nemůže překročit kapacitu hrany
2. $\sum_{xv \in E} f(xv) = \sum_{vx \in E} f(vx), \forall v \in V / \{s,t\}, x \in V$ - co do vrcholu „přiteče“, musí také odtéct

Průtok hranou *e* je značen jako *f(e)* a velikost toku $|f|$ je množství toku, které protéká ze zdroje do spotřebiče, tudíž vzhledem k podmínkám platí rovnost (3.2), kde $f(t)$ tok přitékající do spotřebiče a $f(s)$ je tok vytékající ze zdroje.

$$|f| = f(t) = f(s) \tag{3.2}$$

Nyní se mohu pustit do grafových řezů. Řezem určeným množinou $R \subseteq V$ nazveme množinu orientovaných hran $\delta(R) = \{vw \in E | v \in R \wedge w \in \bar{R}\}$, kde $R \subseteq V$ a $\bar{R}$ je doplňkem. Řez je (s,t)-řez, pokud $s \in R \wedge t \notin R$ a kapacita řezu je $c(\delta(R)) = \sum_{e \in \delta(R)} c(e)$. Velikost každého *(s,t)*-řezu je horním odhadem na velikost toku. Pro názornost uvedu příklad, kdy mám síť (obrázek č. 2), značení u hran znamená „*f(e)/c(e)*“. V této síti je velikost toku 3, velikost řezu určeného vrcholy $\{s, 1,2\}$ je 6 a velikost řezu $\{3,4, t\}$ je 1.



Obrázek 2: Příklad toku grafem

Po definování pojmů začínají problémy. V případě toků v sítích jde o problémy dva a to problém hledání maximálního toku a minimálního řezu (řezu nejmenší možné kapacity). Nebudu se zde příliš zabývat důkazy a tak pouze uvedu větu objevenou Fordem a Fulkersonem. [9] Jde o větu o maximálním toku a minimálním řezu, která říká, že pokud existuje maximální *(s,t)*-tok, pak platí rovnice (3.3). Jinak řečeno, maximální tok je roven minimálnímu řezu. Existence minimálního řezu je lehce dokazatelná, neboť počet řezů v každé síti je konečný a toky tvoří v prostoru všech ohodnocení hran kompaktní množinu a velikost toku je lineární spojitá funkce a tudíž nabývá maxima. Algoritmy řešící minimální řez a maximální tok jsou například Ford-Fulkersonův, Dinicův/Edmonds-Karpův, Goldbergův Push-Relabel a další.

$$\max\{|f|: f\ je\ (s,t) - tok\} = \min\{c(\delta(R)): \delta(R)\ je\ (s,t) - \check{r}ez\} \tag{3.3}$$

3.4.1 Ford-Fulkersonův algoritmus

Jde o jeden z nejstarších algoritmů zabývající se maximálním tokem. Je pojmenován podle L. R. Forda, Jr. a D. R. Fulkersona, kteří jej navrhli v roce 1962. Tento algoritmus patří do skupiny algoritmů augmenting path – algoritmů zlepšující cesty a používá rezervu hrany.

Myšlenka algoritmu je taková, že na začátku máme nulový tok a postupně hledáme cestu, po které bychom dosáhli většího průtoku ze zdroje do cíle, dokud existuje nějaká zlepšující cesta.

Popis algoritmu pseudokódem podle [1]:

$G = (V, E)$, kde ***G*** je graf, ***V*** je množina vrcholů, ***E*** je množina hran

$\forall(u,v) \in E: f(u,v) \leftarrow 0, f(v,u) \leftarrow 0$ … nastavení aktuálního toku grafem na 0

Konstrukce grafu ***G´*** s tokem 0

```
while(existuje zlepšující cesta p v grafu G´ mezi zdrojem a cílem)
{
      cf(p) = min{c´(u,v):(u,v)∈p} … kapacita zlepšující cesty
      foreach((u,v)∈p)
      {
            f(u,v) ← f(u,v) + cf(p)
            f(v,u) ← f(u,v)
      }
      Upravení grafu G´ podle nového toku
}
```

3.4.2 Dinicův/Edmonds-Karpův algoritmus

Tento algoritmus je vylepšením algoritmu Ford-Fulkerson, který zlepšuje časovou složitost pomocí vyhledání nejkratší vylepšující cesty. Tuto možnost uvedli jako heuristiku už Ford s Fulkersonem, ale prozkoumána byla až v roce 1970 Dinicem a nezávisle na něm v roce 1972 i dvojicí Edmonds a Karp. Dinic ještě přišel s vylepšením pomocí vrstevnaté (čisté) sítě a blokující tokem.

Popis algoritmu Dinic podle [5]:

1. Začneme s libovolným tokem *f*, například prázdným (všude nulovým).
2. Iterativně vylepšujeme tok pomoci zlepšujících toků: (vnější cyklus)
3. Sestrojíme síť rezerv: vrcholy a hrany jsou tytéž, kapacity jsou určeny rezervami v původní síti. Dále budeme pracovat s ní.
4. Najdeme nejkratší *st*-cestu. Když žádná neexistuje, skončíme.
5. Pročistíme síť, tj. ponecháme v ni pouze vrcholy a hrany na nejkratších *st*-cestách.
6. Najdeme v pročištěné síti blokující tok f_B:

7. $f_B \leftarrow$ prázdný tok
8. Postupně přidáváme *st*-cesty: (vnitřní cyklus)
   9. Najdeme *st*-cestu. Např. hladově - „rovnou za nosem“.
   10. Pošleme co nejvíce po nalezené cestě.
   11. Smažeme nasycené hrany. (Pozor, smazáním hran mohou vzniknout slepé uličky, čímž se znečistí síť a nebude fungovat ladové hledání cest.)
   12. Dočistíme síť
13. Zlepšíme *f* podle f_B

Blokující tok je takový tok, že každá orientovaná *st*-cesta obsahuje alespoň jednu nasycenou hranu.

3.4.3 Push-Relabel algoritmus

[1] Tento algoritmus přichází s odlišným přístupem než algoritmy vylepšující cesty. Ideu tohoto algoritmu lze popsat příkladem, kdy graf „pověsíme“ za zdroj a necháme viset kolmo dolů tak, že se spotřebič dotýká země a necháme protéct například vodu ze zdroje do spotřebiče. Z každého vrcholu poteče voda, která přebývá, do vrcholu v menší výšce a množství vody, která proteče, je omezeno kapacitou hran mezi těmito vrcholy. Takto pokračujeme a sledujeme vodu do spotřebiče. Postup ukončíme, jakmile v žádném uzlu nebude přebývat voda.

Nejdelší cesta od zdroje ke spotřebiči obsahuje maximálně *V* vrcholů, takže výšku každému vrcholu přiřadíme následujícím způsobem:

- Výška(s) = V
- Výška(t) = 0
- $f(u, v) > 0$ jestliže Výška(u) > Výška(v)

Takže výška vrcholů je postupně upravována a tok protéká mezi vrcholy, dokud maximální tok nedosáhne spotřebiče. Následně pokračuje zvyšování výšek vrcholů uvnitř grafu, dokud se všechen tok, který nedorazil do spotřebiče, nevrátí zpět do zdroje. Pro zavedení pojmu *přebytek (excess)*, je třeba nejdříve definovat pojem *vlna (preflow)*. Vlna je podobná toku, ale s rozdílem, že vlna povoluje kladný přebytek ve vrcholu grafu.

Preflow *P* je tok pokud:

- $\forall(u, v) \in E: P(u, v) \leq c(u, v)$
- $\forall(u, v) \in E: P(v, u) = -P(u, v)$
- $\forall(u) \in V - \{s\}: \sum_{(w,u)\in E} P(w, u) - \sum_{(w,u)\in E} P(u, w) \geq 0$

Přebytek ***e*** vrcholu ***u*** je dán jako $e(u) = P(V, u)$, což znamená, že přebytek vrcholu je dán rozdílem mezi přitečeným a odtečeným tokem z vrcholu *u*.

Obecná implementace Push-Relabel algoritmu podle Goldberga a Tarjana z roku 1988 [6] vypadá následovně:

```
while(pokud existuje vrchol v s přebytkem (e(v)>0))
{
```

```
        Vyber vrchol v.
        Udělej operaci Push (Push(v)) nebo
        udělej operaci Relabel (Relabel(v))
}
```

Operace *Push()* provádí odstranění přebytku z vrcholu. K provedení operace, kdy bude poslán tok rovný menší hodnotě z dvojce *(e(u),c(u,v)-f(u,v))*, musí být splněny následující podmínky:

- *e(u)* > 0 : ve vrcholu *u* je přebytek
- *c(u,v) – f(u,v)* > 0 : kapacita hrany *(u,v)* není naplněna
- *výška(u) > výška(v)* : vrchol *u* má větší výšku než vrchol *v*

Operace *Relabel()* provádí zvětšování výšky vrcholu grafu, dokud není vyšší než alespoň jeden z vrcholů, který má volnou kapacitu. Podmínky pro provedení operace jsou:

- *e(u)* > 0 : ve vrcholu *u* je přebytek
- *výška(u) ≤ výška(v)* pro $\forall v: c(u,v) = f(u,v) > 0$: Všechny vrcholy které mají volnou kapacitu a vede do nich hrana z *u* mají větší výšku

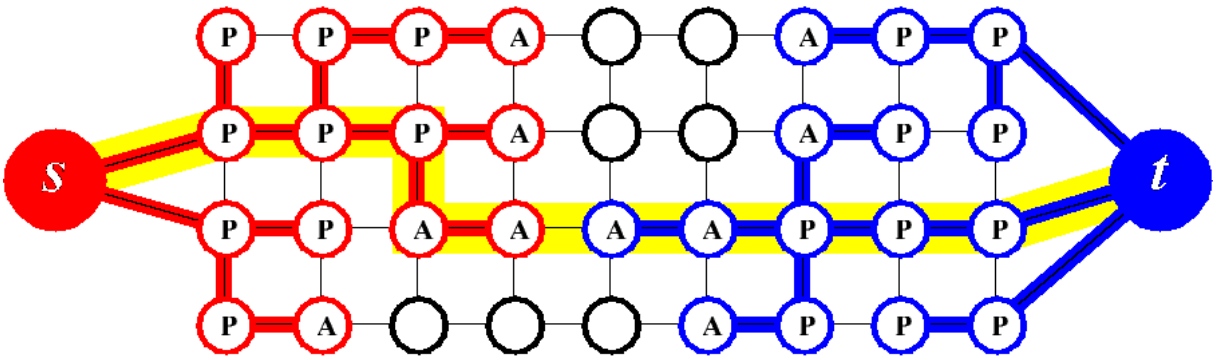
Při operaci *Relabel()* je zvětšena výška vrcholu o nejmenší možnou hodnotu tak, aby platila nerovnost *výška(u) > výška(v)* pro všechny vrcholy *v* mající $c(u,v) - f(u,v) > 0$.

3.4.4 Vylepšený min cut/max flow algoritmus podle V. Kolmogorova a Y. Boykova pro pixel labeling problem [8]

Kamenem úrazu u algoritmů pro vyhledání minimálního řezu/maximálního toku typu vylepšující cesty je vytváření nového vyhledávání do šířky (breadth-first search) pro s-t cesty jakmile jsou vyčerpány všechny cesty dané délky. Při použití v oblasti počítačového vidění a zpracování obrazu znamená operace vytvoření vyhledávacího stromu pro breadth-first search zpracovat velkou množinu pixelů ze zpracovávaného obrazu, což je drahá operace, zvláště pokud je nutné vyhledávací strom tvořit příliš často. Takže vylepšení V. Kolmogorova a Y. Boykova se vztahuje právě na tuto oblast.

3.4.4.1 ***Teorie***

Jak již bylo zmíněno výše, algoritmus patří do skupiny algoritmů augmenting paths tedy zlepšující cesty a podobně jako Dinic/Edmonds-Karpův algoritmus vyhledává zlepšující cesty pomocí vyhledávacích stromů. První rozdíl mezi těmito postupy je použití vyhledávacích stromů dvou namísto jednoho, jeden je tvořen od zdroje a druhý od cíle. Rozdíl číslo dva tkví v tom, že vyhledávací stromy nejsou stavěny pro každé vyhledávaní od začátku, ale jsou využívány opakovaně.



Obrázek 3: Vyhledávací stromy algoritmu V. Kolmogorova a Y. Boykova [8]

Obrázek 3 ukazuje příklad řešení algoritmu, kde lze vidět vyhledávací strom S a vyhledávací strom T, které se vzájemně nepřekrývají. Hrany ve stromě T, které vedou od rodičovských uzlů k uzlům dětským, nejsou saturované (nemají zaplněnu svou kapacitu), naproti tomu ve stromě T hrany od dětských uzlů k rodičovským nejsou saturované. Ty uzly, které nenáleží žádnému z grafů, jsou označovány jako volné. Matematickým zápisem vypadá situace jako (3.3). Uzly, které náleží stromu S nebo T, mohou být aktivní nebo pasivní, přičemž aktivní jsou takové, které nemají potomka a pasivní jsou ostatní. Aktivní místa umožňují vyhledávacímu stromu zvětšovat se po nesaturovaných hranách přibíráním volných uzlů. Zlepšující cesta je nalezena ve chvíli kdy jeden ze stromů zjistí, že sousední uzel již není aktivní a patří do druhého stromu.

$$S \subset V, s \in S, T \subset V, t \in T, S \cap T = \emptyset \tag{3.3}$$

Algoritmus se skládá ze tří fází, které se iterativně opakují. Fáze jsou následující:

- „growth“ – fáze růstu, vyhledávací stromy S a T rostou (přibírají volné uzly) dokud na sebe nenarazí a nevznikne cesta mezi zdrojem a cílem.
- „augmentation“ – fáze zlepšení, nalezená cesta vylepšená a z vyhledávacího stromu se stane les (les je takový neorientovaný graf, kdy jsou každé dva uzly spojeny nejvýše jednou hranou)
- „adoption“ – fáze adopce, dojde k obnově stromů S a T.

Fáze „growth“ je místem, kdy dochází k expanzi vyhledávacích stromů. Aktivní uzly prozkoumají přiléhající nesaturované hrany a získají potomky z volných uzlů, které se tak stanou novými aktivními uzly stromu. Uzel se stane pasivním po prozkoumání všech sousedů. Tato fáze končí ve chvíli, kdy aktivní uzel při prohledávání sousedů najde uzel, který již náleží druhému vyhledávacímu stromu, čímž je nalezena cesta od zdroje k cíli (vyznačená žlutá cesta na obrázku 3).

Ve fázi „augmentation“ dochází ke zlepšování cesty nelezené ve fázi „growth“ a to takovým způsobem, že je touto cestou poslán co největší možný tok (od zdroje k cíli), čímž dojde k saturaci některých hran a to vede k situaci, že z některých uzlů se stanou sirotci. Uzel se stává sirotkem v případě, kdy je kapacita jeho hrany s rodičem naplněna, což nevyhovuje počáteční definici o nesaturovaných hranách ve vyhledávacích stromech. Tímto krokem může dojít k rozpadu stromu a

vzniku lesa, kdy kořenové uzly *s* a *t* zůstanou kořenovými uzly dvou stromů a vzniklé sirotčí uzly se stanou kořenovými uzly vzniklých stromů.

Fáze „adoption“ má zajistit odstranění nově vzniklých stromů z fáze „augmentation“ a obnovit tak situaci pouze dvou existujících vyhledávacích stromů S (s kořenem *s*) a T (s kořenem *t*). Způsob jak toho docílit je adopce, najití nového vhodného rodičovského uzlu sirotčímu uzlu. Rodič by měl pokud možno náležet do stejného vyhledávacího stromu jako původně sirotek a měl by být propojen nesaturovanou hranou. V případě, že takový uzel není, stane se sirotek volným uzlem a všichni jeho potomci se stanou sirotky. Pokud nezbývají žádní sirotci, fáze končí.

Po ukončení fáze adopce nastává opět fáze růstu. Celý proces je ukončen ve chvíli, kdy vyhledávací stromy nemohou dále růst a jsou od sebe odděleny hranami se zaplněnou kapacitou. Množina uzlů *S* minimálního řezu grafu je rovna množině uzlů vyhledávacího stromu S, stejně tak to platí i pro množinu uzlů cíle. Maximální tok grafem je dán součtem kapacit hran mezi vyhledávacími stromy.

3.4.4.2 ***Implementace***

Je dán orientovaný graf $G = (V, E)$, kde $\boldsymbol{G}$ je graf, $\boldsymbol{V}$ je množina vrcholů, $\boldsymbol{E}$ je množina hran. V úvahu bereme pouze reziduální graf, to je takový orientovaný graf, kde kapacity všech hran jsou větší než nula. V algoritmu jsou potřeba pomocné množiny a to množina aktivních uzlů *A,* sirotků *O* a vylepšující cesty *P*. Vyhledávací stromy je vhodné reprezentovat pomocí příznaků, kdy příznak *strom(u)* každému uzlu grafu *u* určí hodnotu podle (3.4). Informace o rodičovství bude uložena jako *rodic(u)*, kdy pro volné a sirotčí uzly a kořenové uzly *s* a *t* platí (3.5). Reziduální kapacita hrany mezi *u a v* je uložena jako *cap(u,v)*.

$$strom(u) = \begin{cases} S\ jestliže\ u \in S \\ T\ jestliže\ u \in T \\ \emptyset\ jestliže\ u\ je\ volný \end{cases} \tag{3.4}$$

$$rodic(u) = \emptyset \tag{3.5}$$

```
initialize: S = {s}, T = {t}, A = {s,t}, O = {}, P = {}
while(true)
{
      growth S nebo T a hledej vylepšující cestu P z s do t
      if (P = {}) break
      augmentation P
      adoption O
}
growth
{
      while(A ≠ {})
      {
```

```
            vyber aktivní uzel u ∈ A
            for each(soused v kdy cap(u,v) > 0)
            {
                  if( strom(v)={})
                  {
                        strom(v) = strom(u)
                        rodic(v) = u
                        A = A ∪ {v}
                  }
                  if(strom(v) ≠ {} and strom(v) ≠ strom(u))
                        return P=path_{s→t}
            }
            vyjmi u z A
      }
      return P = {}
}
augmentation()
{
      najdi hranu s nejmenší kapactou Δ na ceste P
      aktualizuj reziduální graf protečením toku Δ po ceste P
      for each(saturovanou hranu (u,v) v P)
      {
            if(strom(u) = strom(p) = S)
            {
                  rodic(v) = {}
                  O = O ∪ {v}
            }
            if(strom(u) = strom(p) = T)
            {
                  rodic(u) = {}
                  O = O ∪ {u}
            }

      }
}
adoption()
{
      while O≠{}
      {
            vyber u ∈ O
            vyjmi u z O
            zpracuj(u)
      }
}
zpracuj(u)
{
      for each(soused v od u)
```

```
        {
            if(strom(v) = strom(u) and cap(v,u)>0 and strom(v)=(S/T))
                rodic(u) = v break
        }
        if(rodic(u)={})
        {
            for each(soused v od u)
            {
                if(strom(v) = strom(u))
                {
                    if(cap(v,u) > 0)
                        A = A u {v}
                    if(rodic(v) = p)
                    {
                        O = O u {v}
                        rodic(v) = {}
                    }
                }

            }
            strom(u) = {}
            A = A - {u}
        }
}
```

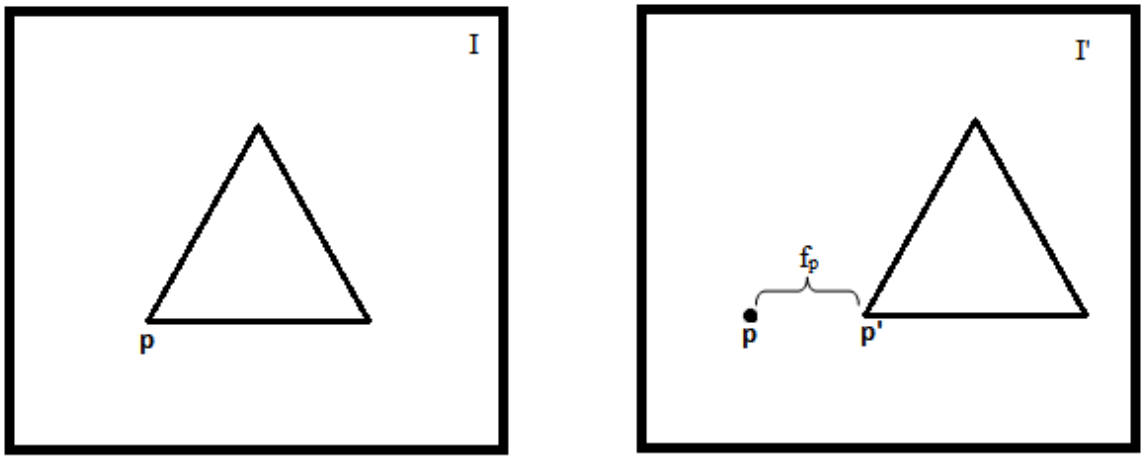
3.5 Pixel labeling problém

Tradičním problémem počítačového vidění je hledání hloubky obrazu s pomocí dvou a více obrazů téže scény, já se v této práci omezím na použití obrazů dvou. Jednou z metod řešení tohoto problému je vyhledání vzájemně si odpovídajících pixelů v obrazech a nazývá se *pixel labeling problem*.

Cílem pixel labeling problému je najít a přiřadit každému pixelu p určitou hodnotu l, která se nazývá *label*. Pixel labeling problém je minimalizační úloha, takže jde o nalezení takového přiřazení, aby byla energie vytvořeného systému co nejmenší. Matematicky definováno pro každý pixel $p = (p_x, p_y) \in P$, kde P je množina pixelů z levého obrazu, najdeme hodnotu l z množiny labelů L. Přičemž hledáme takové ohodnocení $f = (f_1, \dots, f_p, f_{|P|})$, které minimalizuje energetickou funkci (3.6).

$$E(f) = \sum_{p} D_p(f_p) + \sum_{\{p,q\} \in N} V(f_p, f_q) \tag{3.6}$$

Kde D_p je energie za přiřazení labelu pixelu p, V je energie za přiřazení určitého páru labelů sousedním pixelům a jedná se o určení prostorové hladkosti a N je systém sousedů. Cílem je získat takové ohodnocení f, kdy bude $E(f)$ nejmenší.



Obrázek 4: Znázornění labelu pixelu

[4] Předpokládáme, že odpovídající si pixely mají stejnou, či velmi podobnou intenzitu a tak term D_p získáme z rozdílu jejich intenzit na druhou (rovnice 3.7). Term V slouží k zamezení nespojitostí u sousedních pixelů s velmi podobnou intenzitou (rovnice 3.8), kde funkce T má předpis (3.9) a λ je empiricky vybraná klesající funkce taková, že (rovnice 3.10) a parametr λ je vybrán úměrně k parametru K podle předpisu (3.11). K je parametr odhadující množství šumu v obraze. Podmínky pro term V jsou (3.12), (3.13), (3.14). Pokud jsou splněny první dvě podmínky, jedná se o takzvaný semi-metrický term, pokud je splněna i třetí podmínka V term metrický.

$$D_p = \left\| I(p) - I'(p + f_p) \right\|^2 \tag{3.7}$$

$$V(f_p, f_q) = \lambda_{p,q} T[f_p \neq f_q] \tag{3.8}$$

$$T[b] = \begin{cases} 1 \; pokud \; je \; b \; pravda \\ 0 \; jinak \end{cases} \tag{3.9}$$

$$\lambda_{p,q} = \begin{cases} 3\lambda \; pokud \; \Delta I(p,q) < 5 \\ \lambda \; jinak \end{cases} \tag{3.10}$$

$$\lambda = \frac{K}{2} \tag{3.11}$$

$$V(\alpha, \beta) = 0 \Leftrightarrow \alpha = \beta \; nebo \; V(\alpha, \beta) \neq 0 \; \Leftrightarrow \; \alpha \neq \beta \tag{3.12}$$

$$V(\alpha, \beta) = V(\beta, \alpha) \geq 0 \tag{3.13}$$

$$V(\alpha,\beta) \leq V(\alpha,\gamma) + V(\gamma,\beta) \tag{3.14}$$

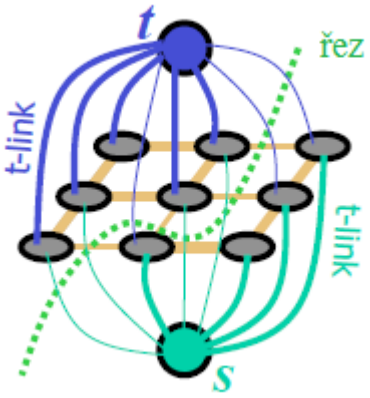
3.6 Pixel labeling problem a grafové řezy

Jak již bylo řečeno, tak pixel labeling problem je problémem minimalizačním, čili hledáme takové ohodnocení, při kterém dostáváme nejmenší energii. U pixel labeling problému je možné pro tuto minimalizaci využít právě grafových řezů, kdy minimální řez grafu bude roven minimu energetické funkce, ovšem je třeba nejprve vytvořit graf.

3.6.1 Vytvoření grafu

K vytvoření grafu máme k dispozici množinu pixelů, množinu labelů a energetické termy, jejichž energii je potřeba minimalizovat.

Graf je tvořen uzly a orientovanými hranami. Uzly grafu jsou tvořeny pixely a labely a hrany jsou dvou typů: n-link a t-link. Hrany typu n-link spojují sousední páry pixelů a kapacita těchto hran je odvozena z termu 3.6. T-link hrany spojují uzel-pixel s uzlem-label a kapacita hran je odvozena od termu 3.5.

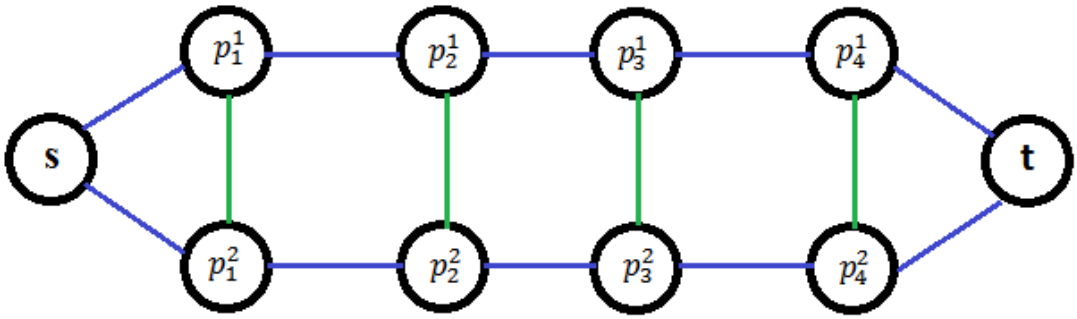


Obrázek 5: Řez grafem

Postup vytvoření grafu je následující [1]:

- Každému pixelu *p* levého obrazu přiřadíme řetěz *k-1* uzlů (p_1, p_2,…,p_{k-1}) a tyto uzly propojíme hranami typu t-link. Hrany označíme jako $\{t_1^p, t_2^p, \dots, t_k^p\}$, kde $t_1^p = (s, p_1)$, $t_j^p = (p_{j-1}, p_j)$ a $t_k^p = (p_{k-1}, t)$. *K* je počet disparit. Kapacita t-link hran je rovna $D_p(l)$.
- Každý pár sousedících pixelů *p, q* spojíme hranou typu n-link. Kapacita n-link hran je rovna $V_{p,q}(l_p, l_q)$.

Obrázek č. 5 zobrazuje graf dvou sousedících pixelů. Množina disparit je {0,1,2,3,4}, takže počet disparit *k=5*. Modré hrany jsou typu t-link a zelené typu n-link.

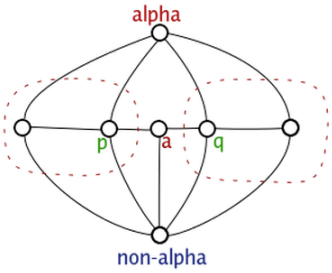


Obrázek 6: Graf pro k=5

3.6.2 Řez grafem a α-expansion move

Pomocí metody grafových řezů je pixel labeling problem řešen velice snadno a rychle například algoritmem Ford-Fulkerson. Minimální řez pak odpovídá ohodnocení, které dává minimální energii. Problémem ovšem je, že tímto způsobem je možné najít pouze lokální minimum. Pro najití silného lokálního minima lze využít algoritmu nazvaného *α-expansion move*.

Použití *α-expansion move* vypadá tak, že v prvním kroku výpočtu jsou vybrány v náhodném pořadí labely α z původní množiny labelů L a v dalších krocích už je toto pořadí neměnné. Následně je pixel labeling algoritmus inicializován způsobem, že všechny pixely mají přiřazenu hodnotu labelu 0. Poté se vypočítá optimální *α-expansion move* (s-t graph cuts) a pokud dojde ke zmenšení energie, dojde k nahrazení labelu, čili pokud dojde při výpočtu s labelem α ke zmenšení energie, pak se původní label nahradí novým lablem α, jinak je ponechán label původní. Postup je opakován tak dlouho, dokud dochází ke snižování energie nebo nedojde k dosažení požadovaného počtu kroků.



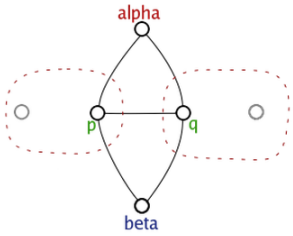
$w(\alpha, p)$	=	$D(\alpha)$
$w(\bar{\alpha}, p)$	=	$D(f_p)$, if $p \notin \mathcal{P}_\alpha$
$w(\bar{\alpha}, p)$	=	∞, if $p \in \mathcal{P}_\alpha$
$w(p, a)$	=	$V(f_p, \alpha)$, and similar for $w(a, q)$
$w(a, \bar{\alpha})$	=	$V(f_p, f_q)$
$w(p, q)$	=	$V(f_p, \alpha)$, when $f_p = f_q$
		$p \in \alpha$ if the link $p - \alpha$ is cut
		$p \in \bar{\alpha}$ if the link $p - \bar{\alpha}$ is cut

Obrázek 7: α-expanze [11]

Po ukončení *α-expansion move* máme minimalizovánu energii pixel labeling problému a tudíž můžeme každému pixelu přiřadit label a vytvořit mapu disparit pixelů, která byla cílem.

3.6.3 α-β swap algoritmus

[11] Hlavní myšlenkou algoritmu α-β swap je oddělit všechny pixely s lablem α od pixelů s lablem β s pomocí grafových řezů. Algoritmus iteruje přes všechny kombinace α – β dokud hodnota výsledné energie nezačne konvergovat.



$w(p,\alpha)$	=	$D(\alpha)+\sum_{\substack{q\in\mathcal{N}_p \\ q\notin\mathcal{P}_\alpha \\ q\notin\mathcal{P}_\beta}} V(f_q,\alpha)$, for $p\in\mathcal{P}_\alpha$ or $p\in\mathcal{P}_\beta$
$w(p,\beta$	=	$D(\beta)+\sum_{\substack{q\in\mathcal{N}_p \\ q\notin\mathcal{P}_\alpha \\ q\notin\mathcal{P}_\beta}} V(f_q,\beta)$, for $p\in\mathcal{P}_\alpha$ or $p\in\mathcal{P}_\beta$
$w(p,q)$	=	$V(\alpha,\beta)$
		$p\in\alpha$ if the link $p-\alpha$ is cut
		$p\in\beta$ if the link $p-\beta$ is cut

Obrázek 8: α-β swap [11]

3.7 Zobrazovací chyby vstupních obrazů

V předchozí části jsem řešil teorii hledání korespondence odpovídajících si pixelů ve dvou vstupních obrazech téměř totožné scény a hledání disparit mezi těmito pixely. Při zkoumání postupů bylo tiše a pro jednoduchost přepokládáno, že rozdíl mezi snímky je pouze v posunutí a čočky pořizovacího zařízení jsou dokonalé. Jinak řečeno, že projekční roviny snímků jsou totožné a na snímcích nedochází k žádnému zkreslení vlivem pořizovacího zařízení. Což s reálnými daty rozhodně často nenastává, a proto je třeba předpřipravit vstupní snímky, tak aby co nejvíce odpovídaly přepokládanému ideálu a použité postupy a algoritmy poskytly co možná nejlepší výsledky.

3.7.1 Chyby optických čoček

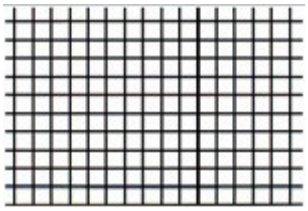
Pokud fotíme třeba čtverec, tak očekáváme, že na výstupním obrazu bude ten čtverec čtvercem, ovšem vlivem optických vad čočky se může stát, že dojde ke zkreslení a linie, kterou jsou při pohledu okem rovné, jsou v obraze nějakým způsobem zakřiveny. K chybám dochází z důvodů nepřesností při výrobě čoček a objektivů. Základními typy zkreslení je soudkovité (radiální) a poduškovité (tangenciální), ale může dojít i ke kombinaci těchto dvou.

Radiální distorze je taková chyba optické čočky, při které dochází k posunu bodu o vzdálenosti r na snímku o hodnotu Δr, čili se vzdáleností od středu snímku roste chyba posunutí bodu. Souřadnice opravených bodů se počítají podle rovnic (3.10), ve kterých se vyskytují parametry

k_1,k_2,k_3. Tyto parametry je potřeba vhodně nastavit, tak aby došlo k co nejlepším výsledkům odstranění distorze na snímcích.

$$x_n = x * (1 + k_1 r^2 + k_2 r^4 + k_3 r^6)$$
$$y_n = y * (1 + k_1 r^2 + k_2 r^4 + k_3 r^6) \quad (3.10)$$
$$r^2 = x^2 + y^2$$

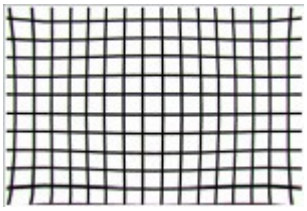
Tangenciální distorze je kolmá na radiální distorzi a je obtížně odstranitelná, reálně ovšem způsobuje jen velice malé deformace, takže je možné toto zkreslení zanedbávat.



Obrázek 9: bez chyby

Obrázek 10: Soudkovité (radiální) zkreslení

Obrázek 11: Poduškovité (tangenciální) zkreslení

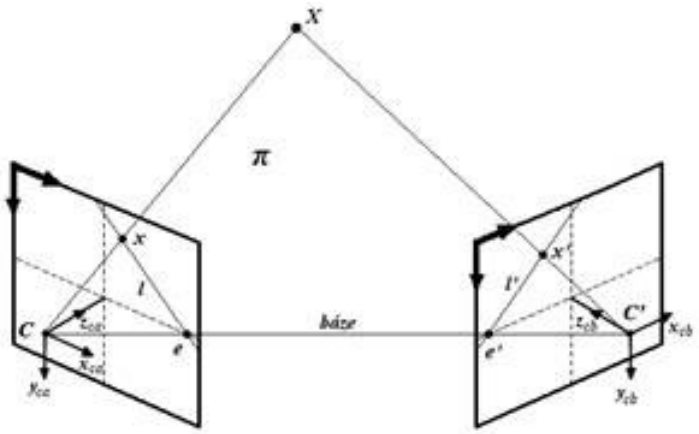


Obrázek 12: Komplexní zkreslení

3.7.2 **Obrazy nejsou snímány v jedné rovině**

Definice pixel labeling problému očekává, že odpovídající si pixely v obrazech leží na stejné přímce, což by znamenalo, že ohniska snímacích zařízení leží v jedné přímce rovnoběžné s projekční plochou. Pěkným příkladem takového systému jsou zdravé lidské oči. V reálných situacích je takovýchto podmínek velice těžké dosáhnout, ať už z důvodu chyby zařízení samotného nebo změny polohy zařízení, ale ve většině případů se takto ani nepostupuje. Reálně jsou snímky vytvářeny z odlišných úhlů a pozic a počítá se s pozdější úpravou snímků pro potřeby rekonstrukce obrazu či jiných odvětví počítačového vidění.

Abychom mohli použít řešení pomocí labeling problému, je třeba provést s obrazy takové transformace, které zajistí, že obrazy budou v jedné rovině. Tento proces se nazývá rektifikace obrazu. Rektifikace je také jedno z velkých témat počítačového vidění a jsou různé způsoby jak dosáhnout rektifikovaných snímků, které jsou rozděleny do dvou skupin – s kalibrací kamer a bez kalibrace. Obě skupiny mají stejný cíl, sestavení fundamentální matice. Jedná se o matici umožňující mapování bodu v jednom obraze na korespondující epipolární přímku v obraze druhém [12]. Pro metody s kalibrací kamer je potřeba znát nebo určitými způsoby zjistit vnitřní vlastnosti kamer a také vnější vztahy kamer mezi sebou. Pomocí těchto informací lze sestrojit transformační matice pro převod snímků, které mají různé souřadné systémy, do jednoho společného souřadného systému. Metody bez kalibrace využívají homografie. Možnostmi jak toho dosáhnout jsou epipolarní geometrie, projektivní rekonstrukce a jiné metody, ale vzhledem k tématu vyhledávání volných parkovacích míst je dostačující projektivní transformace.



Obrázek 13: Epipolární geometrie [13]

3.7.2.1 Epipolární geometrie

Epipolární geometrie je jakousi matkou pro vyhledávání odpovídajících si bodů na snímcích dvou stereo kamer, protože poskytuje potřebný teoretický základ tohoto problému. Epipolární geometrie je vnitřní projektivní geometrie mezi dvěma pohledy. Je nezávislá na struktuře scény, závisí pouze na vnitřních parametrech kamer a na jejich relativní pozici [12]. Na obrázku 13 je vidět v prostoru ležící bod *X*, který snímají dvě kamery s maticemi *P* a *P´*. Na průmětnách těchto kamer je bod *X* promítnut jako *x* a *x´* tak, že platí $x = PX$ a $x' = P'X$. Spojnice středů kamer se nazývá báze a průsečík báze s rovinou průmětny tvoří dipól *e* a *e´*. Spojnice bodů *e* a *x* (*e´* a *x´*) se nazývá epipolární

přímka, na obrázku 13 označena jako l (l'). Důsledkem epipolární geometrie je mapování bodu x jedné projekční roviny na epipolární přímku l' té druhé. Pro každé dva odpovídající si body x a x' platí (3.11) a pro výpočet epipolární přímky platí (3.12), kde F je fundamentální matice [13].

$$x'^{T} F x = 0 \tag{3.11}$$

$$l' = F x \tag{3.12}$$

3.7.2.2 ***Homografie***

Homografie neboli projektivní transformace je invertibilní transformace mezi dvěma projektivními perspektivami, kde zásadní vlastností je mapování přímek opět na přímky. Pro body v homogenních souřadnicích lze toto mapování zapsat rovnicí (3.13), kde H je transformační matice homografie [13]. Pro výpočet matice homografie je potřeba znát minimálně souřadnice čtyř odpovídajících si bodů ve stereo obrazech, kdy žádné tři z těchto bodů neleží v jedné přímce.

$$x'_i = H x_i \tag{3.13}$$

3.7.2.3 ***Rektifikace***

Rektifikace je proces transformace obrazu, při kterém dojde k převedení všech prvků obrazu z jednoho souřadného systému do druhého. Protože v mém případě neznám projekční matice kamer snímků, které mám k dispozici a ani možnost je zjistit, zbývá mi jako metodu pro rektifikaci obrazu zvolit rektifikaci pomocí homografie. Ideálními kandidáty na potřebné čtyři body jsou rohy parkoviště, jsou pravidelně rozmístěny a žádné tři rozhodně neleží v jedné přímce.

4 Programování aplikace

V této kapitole se budu věnovat programování aplikace pro rozpoznávání volných parkovacích míst pomocí metody grafových řezů. Popíši zde knihovny, postupy a důvody proč jsem takové postupy zvolil a také ukážu a okomentuji konečné výsledky.

Aplikace je napsána v programovacím jazyce C++, použitá knihovna pro zpracování obrazu je openCV 2.4.3 a knihovna řešící minimální řez a maximální tok s optimalizací α-expansion nebo α-β *swap* je gco-v3.0.

4.1 Programovací jazyk C++

Při výběru programovacího jazyka jsem se rozhodoval mezi C++ a C#. Jako vždy, když se pouštím do podobného projektu, padla volba na C++. Důvod byl jednoduchý, odpadají problémy s integrací a používání knihoven, protože mají pro C++ stabilní verze, poměrně dlouhou dobu vývoje a spousty uživatelů. Verze pro C# nejsou většinou na světě dostatečně dlouho a nemají tak silnou developerskou a uživatelskou podporu.

4.2 OpenCV

Protože se jedná o diplomovou práci z oblasti počítačového vidění a zpracování obrazu, zvolil jsem pro práci s obrazovými daty knihovnu OpenCV verze 2.4.3. OpenCV je opensource multiplatformní knihovna pro manipulaci s obrazovými daty, obsahující stovky algoritmů pro práci s počítačovým viděním, která je vyvíjena pod BSD licencí.

4.3 gco-v3.0

Gco je knihovna pro optimalizování multi-label energií pomocí α-expansion a α-β-swap algoritmu. Je napsána v programovacím jazyce C++ a je vyvíjena *Computer Vision Research Group* na univerzitě v západním Ontariu.

Verze 3.0 této knihovny obsahuje pět hlavičkových a pět zdrojových souborů (block.h/cpp, energy.h, GCoptimization.h/cpp, LinkedBlockList.h/cpp, graph.h/cpp, maxflow.cpp), které vyvinula Olga Veksler s pomocí implementací od V. Kolmogorova a Y. Boykova. Publikace, ze kterých tyto kódy vychází, jsou [4], [7], [8].

4.3.1 energy.h

Tento kód vytvořil V. Kolmogorov. Podle práce [4] implementoval techniku minimalizace binární energie. V této knihovně je tato technika používána k minimalizaci binární energie pro krok α-expanze a swap algoritmu.

Binární energií jsou myšleny energetické funkce binárních proměnných, které řeší podproblém α -expansion move algoritmu. Tento podproblém je výpočet nejmeší energie labelingu během jednoho kroku α-expanze a je možné jej řešit pomocí binárních proměnných, přestože α-expansion move

algoritmus má na vstupu proměnné, které nenabývají pouze binárních hodnot. Převedení vstupních hodnot na binární je možné, jelikož při α-expanzi si každý pixel buď ponechá původní hodnotu z konfigurace, nebo ji změní na hodnotu α.

Zapsáno formálnějším způsobem, jakýkoliv labeling f' vzniklý během jednoho kroku α-expanze z počáteční konfigurace labelingu f je možné zakódovat pomocí vektoru x s binárními hodnotami tak, že $x = \{x_p | p \in P\}, kde\ f'(p) = f(p)\ jestliže\ x_p = 0\ a\ f'(p) = \alpha\ pokud\ x_p = 1$. Jelikož je funkce energie *E* definovaná nad všemi labelingy, tak je definovaná i nad všemi labelingy binárních vektorů *x*. Takto je potřeba nalézt minimum funkce energie $E(f^x)$ všech binárních vektorů *x* (f^x označuje labeling binárního vektoru *x*).

Výhodou toho přístupu je, že v případě binárních hodnot přiřadí grafový řez každému uzlu grafu efektivněji jednu ze dvou možných hodnot.

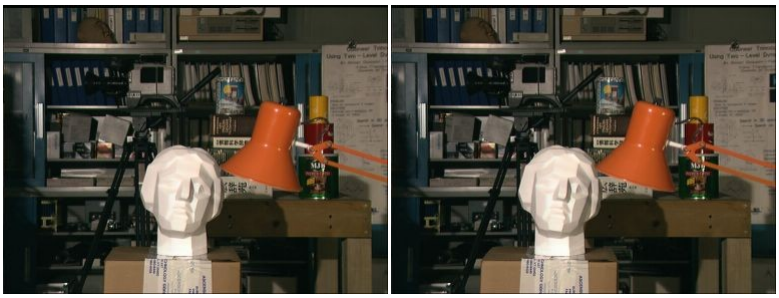
4.3.2 **maxflow.cpp, graph.cpp/h, block.h**

Tyto zdrojové kódy jsou vytvořeny Y. Bykovem a V. Kolmogorovem během výzkumu pro společnost Siemens. Použitý algoritmus vychází ze studie [8] a slouží k vytvoření grafu a výpočtu minimálního řezu/maximálního toku.

Tvůrci vyšli ze skupiny algoritmů zlepšující cesty a snažili se zefektivnit výpočet minimálního řezu/maximálního toku pro potřeby počítačového vidění. Vytvořili postup podobný algoritmu Dinic, který používá k vyhledání zlepšující cesty vyhledávací stromy. Kolmogorov s Bykovem použili vyhledávací stromy dva (jeden vyhledává od zdroje a druhý od cíle). Dalším vylepšením je pak používání již vytvořených vyhledávacích stromů místo jejich vytváření od začátku. Podrobně je algoritmus popsán v kapitole 3.4.4.

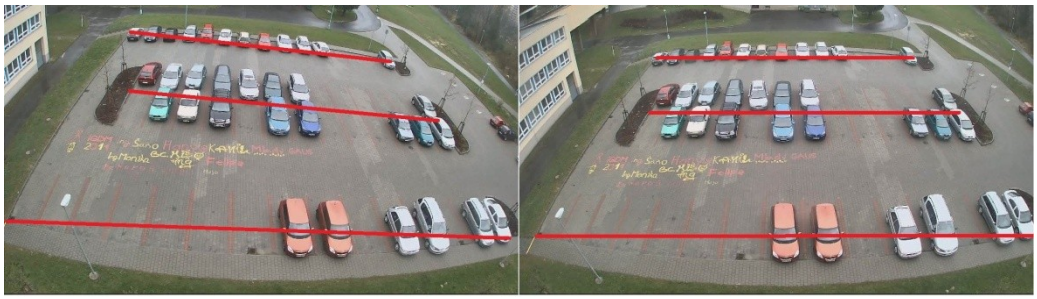
4.4 Rektifikace vstupních obrazů

Teorie řešení pixel labeling problému předpokládá, že při hledání korespondujících si pixelů v levém a pravém obrazu se hledaný pixel nachází vpravo nebo vlevo na stejné úrovni. Jinak řečeno pravý snímek je od levého posunutý o kousek doprava po společné ose *x*, osy *y* a *z* jsou rovnoběžné, například jako v této problematice velice často používané snímky „tsukuba“ (obrázek č. 14). Realita při snímání parkoviště je ovšem zcela jiná (např. obrázek č. 15). Ideální z hlediska řešení pixel labeling problému by bylo, kdyby rovina *xz* kamery byla buď rovnoběžná s parkovištěm, nebo kolmá na parkoviště. V rovnoběžném případě by ovšem bylo vidět pouze první řadu parkovacích míst, tudíž je tato možnost pro rozpoznávání volných míst nevhodná. Kolmý případ by byl pro rozpoznávání optimální, ovšem realizace by byla technicky a finančně náročná. Z příkladu reálné dvojce snímků parkoviště lze na první pohled vidět, že pixely v levém snímku nebudou mít odpovídající pixely ve snímku pravém na stejné úrovni a dokonce jednotlivé pixely v jedné úrovni levého snímku mají svůj odpovídající pixel v pravém snímku na různých úrovních, jak je přibližně znázorněno na obrázku č. 16. Tento problém je potřeba vzhledem k pixel labeling problému vyřešit a to pomocí rektifikace obrazů. Co to je rektifikace obrazů, je zmíněno výše.



Obrázek 14: Tsukuba - levý a pravý snímek

Obrázek 15: Parkoviště - levý a pravý snímek



Obrázek 16: Odpovídající si pixely

Způsobů jak rektifikovat obrazy je vícero, některé ovšem vyžadují kalibraci kamer/y, takže je třeba získat informace o kameře a jejím nastavení, jiné zase znalost přesných globálních souřadnic několika bodů v obraze. V této práci jsem se rozhodl použít pro rektifikaci obrazů perspektivních transformací, kterými obrazy „vyrovnám“ a přitom zachovám co nejvíce z původní informace ve snímcích s co nejmenším přidaným zkreslením. Dále je tímto způsobem možné vykompenzovat zkreslení vzniklé natočením kamery, které způsobuje zvětšení parkoviště v levém snímku.

Knihovna OpenCV obsahuje mnoho funkcí pro transformace 2D obrazu a samozřejmě i pro transformace perspektivní. Pro transformaci obrazu je třeba vytvořit transformační matici, v tomto případě perspektivní transformační matici. K vytvoření této matice poslouží openCV funkce `cvGetPerspectiveTransform(src, dst, mapMatrix)`, kde parametr *src* obsahuje souřadnice čtveřice bodů původního obrazu, parametr *dst* obsahuje souřadnice čtyř bodů odpovídajících bodům ze *src*, ale po transformaci a *mapMatrix* je ukazatel na výslednou transformační matici. Teď už je potřeba najít souřadnice čtyř vhodných bodů a souřadnice jím odpovídajících po transformaci. Výběr vhodných bodů záleží na konkrétním případě parkoviště a způsobu pořízení stereo snímků, proto se dále budu zabývat snímky, které mám k dispozici. Vybrat vhodné body není v tomto případě velký problém, na první pohled se naskýtají rohy parkoviště, ve snímcích budou stále na stejném místě a reálně tvoří pravoúhlý čtyřúhelník. Zjištění souřadnic také není problém, stačí

snímek zobrazit pomocí programu, který zobrazuje souřadnice (např. mspaint, GIMP), najít rohy parkoviště a vyčíst souřadnice. Zjištění souřadnic bodů jím odpovídajících po transformaci je o něco náročnější. První myšlenkou by mohlo být například namapování bodů z levého snímku na body z pravého. Tento postup by vedl k přílišné deformaci levého snímku a hlavně by byla ztracena informace stereo obrazu (obrázek č. 17). Body je třeba zvolit tak, aby po transformaci došlo k co nejmenší deformaci snímku (v tomto případě se větší deformaci levého snímku nelze vyhnout, jelikož roviny obou snímků vzhledem ke globálnímu systému jsou docela rozdílné). Nejmenší deformace bude dosaženo v případě, že body původní a po transformaci budou sobě co nejblíže. Zároveň ale je třeba body zvolit tak, aby jejich spojnice tvořící přední a zadní hranu parkoviště byly rovnoběžné. Postup získání souřadnic bodů jsem zvolil následující. V prvním kroku jsem nechal *x*-ové souřadnice z původních snímků a *y*-ové souřadnice horních dvou bodů jsem dal jako průměrnou hodnotu *y*-ových souřadnic původních horních bodů, stejně tak u dolních. Ve druhém kroku už šlo jen o srovnání délek stran parkoviště v obou snímcích. Z najitých souřadnic vytvoří funkce cvGetPerspectiveTransform() perspektivní transformační matici, pomocí které provede funkce `cvWarpPerspective()` perspektivní transformaci na všechny pixely snímku, čímž dosáhneme požadované rektifikace levého a pravého snímku (obrázek č. 18). Na snímcích je možné vidět, že jednotlivé linie odpovídajících si pixelů nejsou v jedné rovině, tento zbývající problém bude dořešen dále v programu, při výběru pixelů „podezřelých“ z korespondence.

Obrázek 17: Namapování bodů levého snímku na body pravého

Obrázek 18: Levý a pravý snímek po rektifikaci

4.5 Odstranění distorze snímků

Snímky parkovišť mohou být deformovány optickými vadami čočky snímacího zařízení (Obrázek č. 19) a mohou nastat situace, kdy je potřeba tuto deformaci odstranit. Ideálním postupem je nejdříve zjistit jakých a jak velkých chyb se dopouští konkrétní zařízení, pomocí focení předmětů o kterých víme, jak jsou veliké a jaké mají vlastnosti. Vhodným předmětem je například velká

šachovnice, ve které je spousta známých a lehce vyhledatelných určujících bodů. Navíc pro kalibraci kamery pomocí šachovnice obsahuje knihovna openCV vhodné funkce a lze najít i mnoho návodů. V případě, že není možné udělat kalibraci předem je potřeba postupovat jiným způsobem. Pro co nejlepší odstranění distorze je potřeba se zaměřit na odstranění nejčastěji se vyskytující, nejvíce deformující a viditelné distorze, kterou je ta radiální.

Obrázek 19: Distorze levého a pravého snímku

Odstranění radiální distorze je možné pomocí rovnic zmíněných v kapitole o vadách optických čoček (rovnice 3.10). Problémem jsou ovšem parametry k_1,k_2,k_3, o kterých bez použití předchozí kalibrace snímacích přístrojů nic nevíme. Možnou metodou pro určení hodnot těchto konstant je metoda „pokus, omyl“, přičemž konstanty nabývají hodnot z intervalu $\langle 0,1 \rangle$. Společně s posunováním pixelů na správná místa je třeba použít interpolaci, aby nedocházelo k rozbití snímku, například bipolární interpolaci. Protože v okrajích snímků je chyba posunutí pixelů velká, dojde při posunu na jejich správnou pozici k tomu, že se nová pozice nachází mimo původní obraz, proto je třeba snímek zvětšit tak, aby nedošlo ke ztrátě informace v okrajích snímků.

Obrázek 20: Odstranění radiální distorze levého snímku

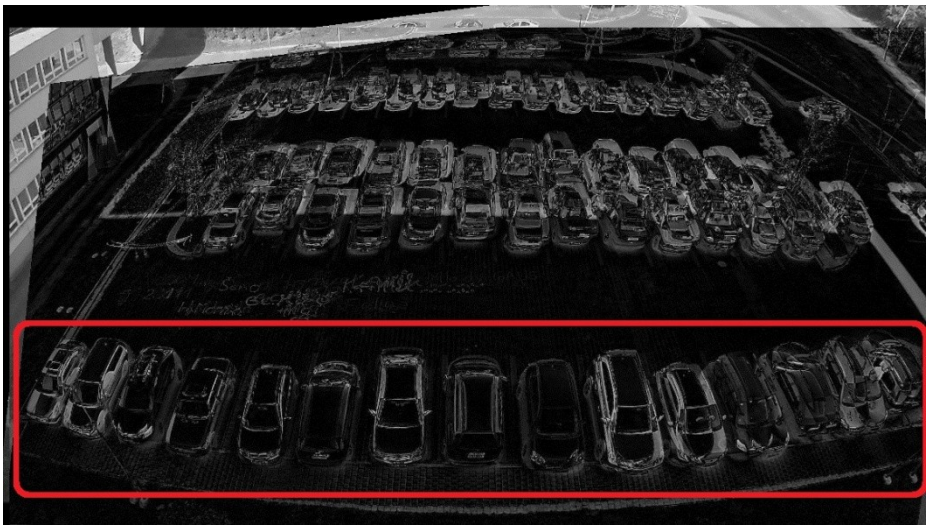
Obrázek 21: Odstranění radiální distorze pravého snímku

4.6 Vytvoření mapy disparit

Nyní se po přípravě snímků dostávám k první z klíčových částí práce, vytvoření mapy disparit ze dvou předem upravených stereo snímků. To znamená vybrat pixely, které mohou ve snímcích korespondovat, z nich vytvořit graf, najít minimální řez a maximální tok grafem, získat disparity jednotlivých pixelů a z disparit vytvořit mapu. Pro usnadnění práce je ještě vhodné převést snímky do odstínu šedi, lépe se pracuje s intenzitami pixelů.

Výběr množiny korespondujících pixelů v levém a pravém snímku není tak snadný jak by se mohlo na první pohled znát. První věcí, kterou je třeba řešit je různá disparita v rámci osy *y* na snímcích, jinými slovy očekávaná disparita korespondujících pixelů bude v dolní části snímků jiná, než v horní části. Druhá věc související s první je, že není třeba prohledávat a zkoušet úplně všechny pixely, protože by to bylo jednak zbytečné a hlavě velice časově náročné na výpočet. Třetí problém k vyřešení je vidět na obrázku č. 22, který zobrazuje rozdíl pravého a levého snímku. Na tomto obrázku je možné vidět, že i po provedené rektifikaci nejsou linie odpovídajících si pixelů na stejné úrovni, takže je třeba provádět hledání korespondujících bodů v pravém snímku na jiných úrovních, než jsou body ve snímku levém. Po posunutí snímků tak, aby se co nejvíce překrývaly (obrázek č. 23) a jejich odečtení, je možné vidět (v červeném obdélníku), že zatímco auta ve středu snímku jsou překryta, tak auta po stranách se nepřekrývají a to jak v podélném směru tak i svislém. To znamená, že je třeba počítat s tím, že v závislosti na poloze vzhledem k ose *x* se mění disparity a zároveň je třeba vyhledávat korespondující pixely na jiných úrovních. Všechny problémy z tohoto odstavce se dají řešit pomocí podélného i horizontálního rozdělení levého snímku a pro jednotlivé vzniklé části nastavit kde v pravém snímku hledat množinu bodů, které by mohly odpovídat pixelům v levém snímku. Čímž se zároveň redukuje mohutnost množiny potřebných labelů.

Obrázek 22: Rozdíl levého a pravého snímku s distorzí



Obrázek 23: Rozdíl levého a pravého snímku s posunutím a s distorzí

Dříve než se začne tvořit graf, je třeba ještě dořešit hodnoty pro *smoothCost* term. To znamená podpořit prostorovou hladkost výsledné mapy disparit pomocí určení sousedních pixelů, které zobrazují stejný objekt na snímku. V praxi to s využitím knihovny *gco-v3.0* znamená vyplnit dvě pole (jedno pro sousedy napravo a druhé pro sousedy pod zkoumaným pixelem) hodnotami podle předpisu z kapitoly 3.5. Tento předpis funkce obsahuje ale dvě ne zcela jasné hodnoty a to práh, který určuje, jak moc odlišné intenzity pixelů zobrazují ještě ten samý objekt v obraze a konstantu *K* odrážející zašuměnost snímku. Obě hodnoty je třeba určit tak, aby pro konkrétní případ použití dávaly co nejlepší výsledek. Praktickými zkušenostmi jsem zjistil, že obě konstanty je třeba pro jednotlivé úrovně obrazu nastavovat zvlášť, auta, která parkují na vzdálenější straně, mají jinou hodnotu šumu než auta parkující blíže.

Po určení hodnot termu *smoothCost* je třeba získat hodnoty pro term *dataCost*. Tyto hodnoty jsou získány jako D_p v kapitole 3.5 pro všechny pixely „podezřelé“ z korespondence k danému pixelu. Protože každému pixelu v levém snímku je třeba přiřadit množinu hodnot, tak výsledné pole může dosáhnout takové velikosti, že nastanou problémy s pamětí. V takovém případě je třeba snímek rozdělit a zpracovat po částech, což je vhodné i pro případnou paralelizaci.

Po spočtení všech hodnot *dataCost* a *smoothCost* knihovna *gco-v3.0* provede vytvoření grafu, najde minimální řez a pomocí *α-expansion move* nebo *α-β swap* algoritmu optimalizuje výslednou energii. Poté má každý pixel levého snímku přiřazen label, ze kterého se vytvoří mapa disparit (obrázek č. 24).

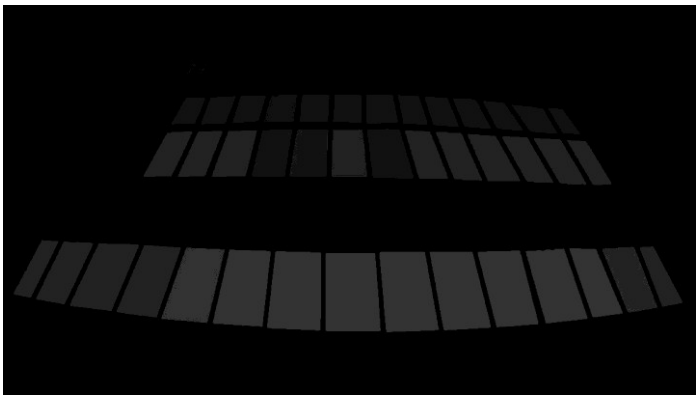
Obrázek 24: Levý snímek s mapou disparit

4.7 Vyhledávání volných/zaplněných parkovacích míst

Teorie vyhledání volných parkovacích míst je taková, že plocha volného parkovacího místa má mít konstantní hodnotu, nejlépe kdekoliv v obraze stejnou a na obsazeném parkovacím místě budou disparity rozdílné. Důležitou věcí, kterou je potřeba znát k rozpoznání zaplněnosti parkovacích míst, je umístění a rozpoznání jednotlivých parkovacích míst. Dále pak najít způsob jak rozpoznat stav parkovacího místa a na konec označit místa v původním snímku, která jsou volná a která obsazená.

Problém jak najít oblast v mapě disparit, kterou prozkoumat jsem řešil vyznačením si parkovacích míst na pomocném levém rektifikovaném snímku bílými obdélníky s černým obsahem (obrázek č. 25). Pomocí převodu do odstínů šedi, prahování, dilatace a eroze jsem docílil černého obrázku s bílými obdélníky reprezentujícími parkovací místa, které je možné oindexovat (obrázek č. 26) a taky pro názornost vyplnit hodnotami z mapy disparit (obrázek č. 27).

Obrázek 25: Vyznačení parkovacích míst na parkovišti

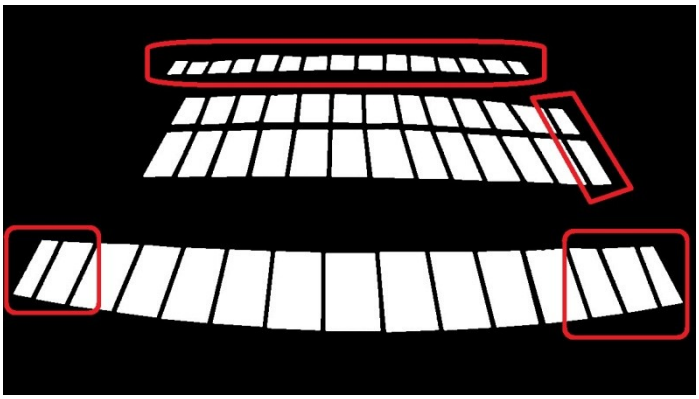


Obrázek 26: Oindexovaná parkovací místa

Obrázek 27: Disparity importované na parkovací místa

Posledním problémem je najít způsob rozhodovaní, zda je parkovací místo plné nebo prázdné. Teorie říká, že obsazené místo má menší disparitu než prázdné. Tato strategie ale nebyla příliš úspěšná, protože nebylo možné najít vhodnou hranici, která by o obsazenosti rozhodovala s většími úspěchy. Jako metodu rozpoznávání jsem tedy zvolil poměr počtu změn disparit na parkovacím místě k celkovému počtu pixelů parkovacího místa. Slabým místem této metody jsou případy, kdy parkovací místo obsahuje malý počet pixelů, což u snímků, které mám k dispozici, nastává při krajích snímku a hlavně na vzdálenější straně parkoviště (obrázek č. 28), kde jsou případná auta, už tak malá vlivem vzdálenosti od snímače, snímána pod tak velkým úhlem, že pixely, na kterých je auto zobrazeno, jsou z velké části mimo parkovací plochu. V případě krajních míst bližší řady dochází zase k tomu, že vlivem úhlu, pod kterým jsou auta snímána, zasahují pixely auta v jednom parkovacím místě do vedlejšího. Všechny tyto problémy mají vliv na výslednou úspěšnost rozhodnutí o zaplněnosti a v těchto problémových místech dochází k nejčastějším chybám. Jak je to s auty na parkovacích místech je vidět na obrázku č. 29. V této metodě je také potřeba nastavit vhodný práh, kdy už je počet změn disparit dostatečně velký a tudíž je předpoklad, že je parkovací místo zaplněné. Opět se při praktické realizaci ukázalo, že pro jednotlivé úrovně je vhodné zvolit práh odlišný.

Poslední zbývající částí už je jen zobrazení parkovacích míst, která jsou volná nebo obsazená. Jeden z možných způsobů může vypadat třeba jako na obrázku č. 30. Tento krok je ovšem čistě jen pro informační účel této práce, pro praktické využití systému pro pomáhání s parkováním je třeba držet informaci o stavu parkovacích míst v seznamu s identifikátorem parkovacího místa a příznakem volno/obsazeno.



Obrázek 28: Problematická místa

Obrázek 29: Auta na parkovacích místech

Obrázek 30: Označení obsazených parkovacích míst

4.8 Úspěšnost rozpoznání obsazených parkovacích míst

Úspěšnost rozpoznání obsazenosti parkoviště pomocí metody grafových řezů ze stereo obrazů je závislá především na pozici parkovacího místa. Chyby nastávají v drtivé většině případů ve vzdálenější řadě parkovacích míst z důvodů zmíněných v kapitole 4.7. Na testovaných příkladech nastalo 0 až 4 chyby na 56 parkovacích míst a pouze jednou se vyskytla chyba mimo nejvzdálenější řadu. Pokud by bylo parkoviště snímáno pod jiným úhlem a tak, že by krajní místa nepodléhala

takovému zkreslení a překrývání (odstranění distorze objektivu nepřineslo prakticky žádné zlepšení úspěšnosti, ale časová náročnost se zvýšila výrazně) předpokládám, že by úspěšnost vzrostla.

4.8.1 Statistiky

Kvůli výše uvedeným důvodům jsou testy na detekci volných parkovacích míst rozděleny po jednotlivých parkovacích řadách a to konkrétně na řady tři. Řady jsou číslovány odshora dolů. Všechny testy proběhly se stejným nastavením aplikace i systému. Snímky parkoviště mají rozlišení 1152x648. Obrázky ke statistikám řady se skládají ze šesti částí:

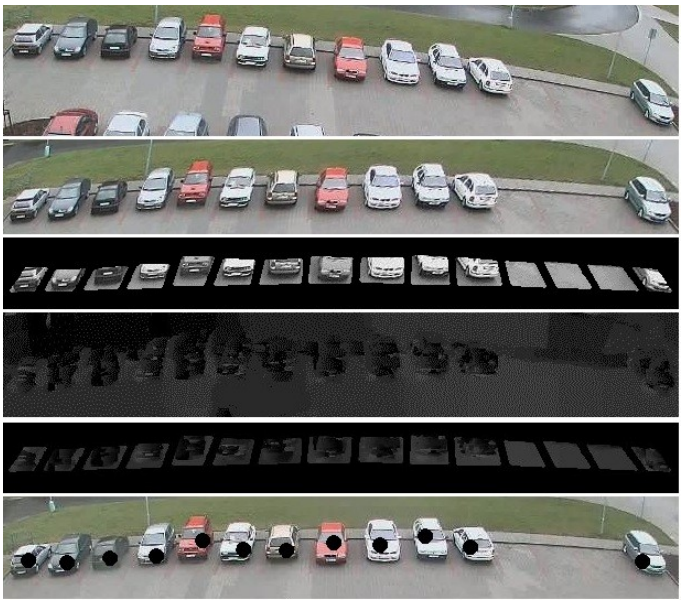
1. Snímek bez jakýchkoliv úprav.
2. Snímek po rektifikaci.
3. Jak se překrývají pixely aut a parkovacích míst
4. Mapa disparit řady
5. Disparity na parkovacích místech
6. Zobrazení výsledku detekce obsazenosti parkovacích míst

4.8.1.1 Snímek parkoviště číslo 1

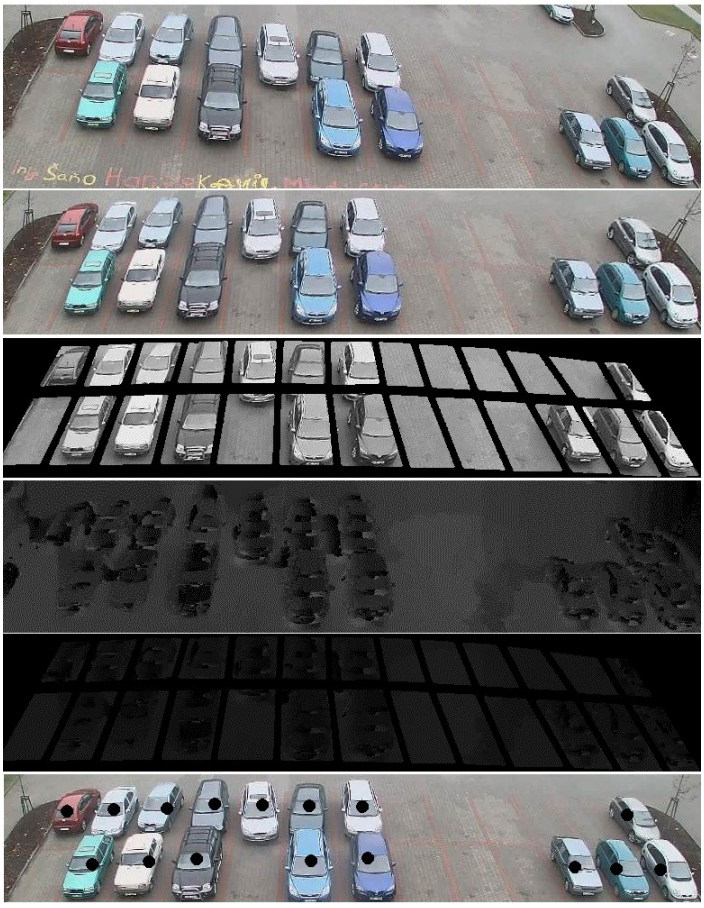
Na tomto snímku probíhá detekce v pořádku, jelikož plocha parkoviště má konstantní charakter, neobsahuje žádné mokré skvrny ani stíny.

Tabulka 1: Statistika detekce parkovacích míst snímku 1

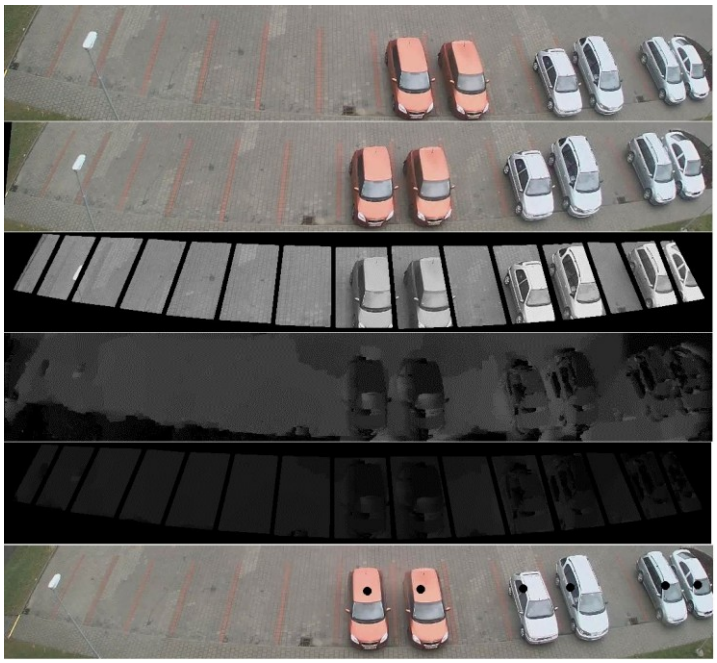
řada	Počet míst	Počet obsazených míst	Počet detekovaných míst	Počet chybně detekovaných míst	Procentuální úspěšnost (%)	Číslo obrázku
1	15	12	12	0	100	31
2	26	16	16	0	100	32
3	15	6	6	0	100	33



Obrázek 31: Parkoviště č. 1, řada první



Obrázek 32: Parkoviště č. 1, řada druhá



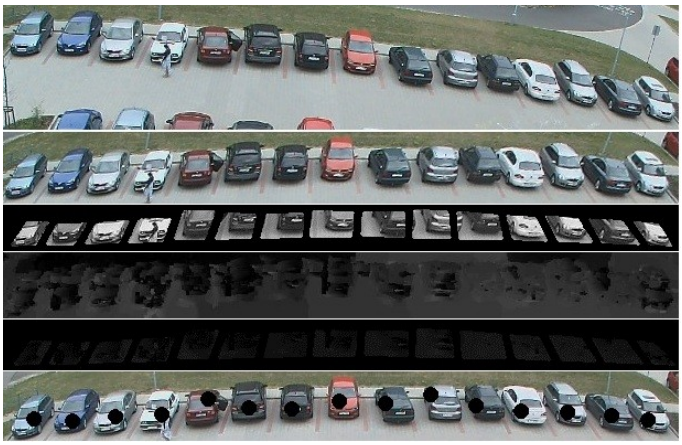
Obrázek 33: Parkoviště č. 1, řada třetí

4.8.1.2 ***Snímek parkoviště číslo 2***

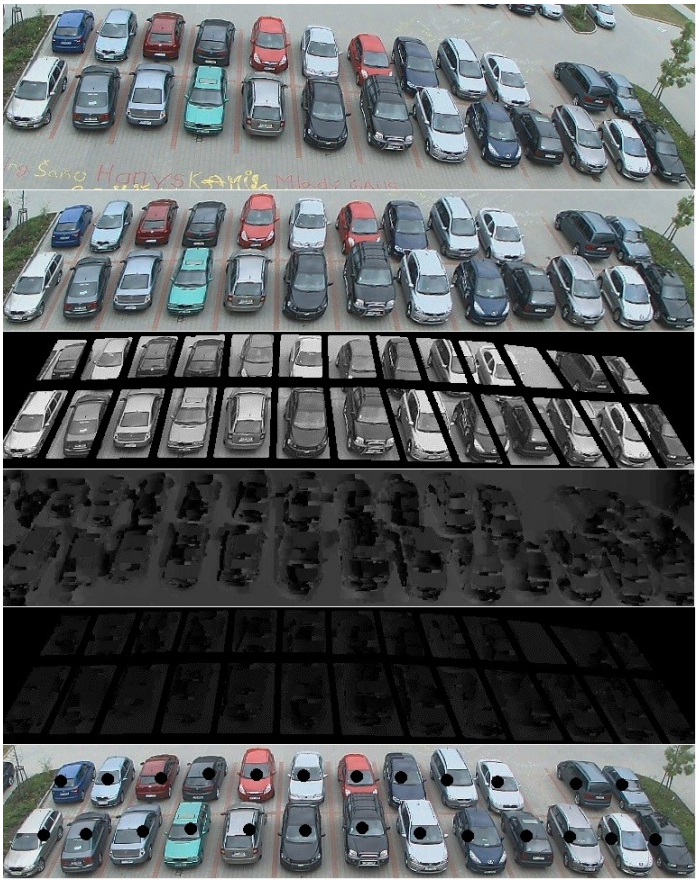
Na tomto snímku probíhá detekce v pořádku, jelikož plocha parkoviště má konstantní charakter, neobsahuje žádné mokré skvrny ani stíny. Přesto jsou některá auta v první řadě blízko hranice rozdělující obsazenost/volnost.

Tabulka 2: Statistika detekce parkovacích míst snímku 2

řada	Počet míst	Počet obsazených míst	Počet detekovaných míst	Počet chybně detekovaných míst	Procentuální úspěšnost (%)	Číslo obrázku
1	15	15	15	0	100	34
2	26	25	25	0	100	35
3	15	14	14	0	100	36



Obrázek 34: Parkoviště č. 2, řada první



Obrázek 35: Parkoviště č. 2, řada druhá

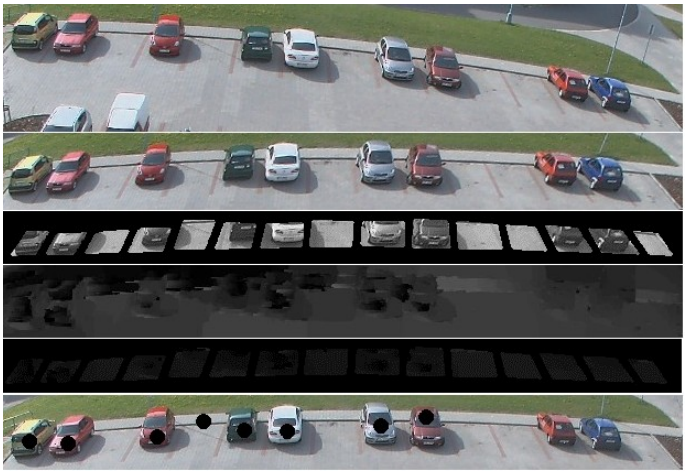
Obrázek 36: Parkoviště č. 2, řada třetí

4.8.1.3 ***Snímek parkoviště číslo 3***

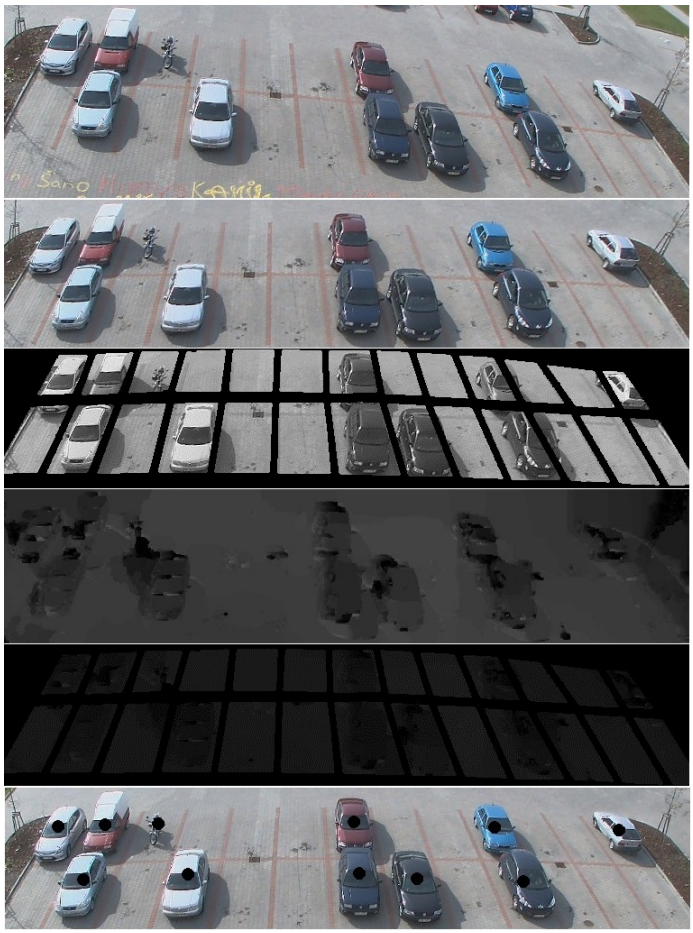
Tento snímek již obsahuje stíny. Na tomto snímku je zaparkovaná motorka, což je zajímavý případ pro testování, protože zabírá malou plochu parkovacího místa. V první řadě jsou chybně detekována 3 místa, jedno volné místo je detekováno jako plné (z důvodu kombinace špatného zaznačení parkovacího a vlivem stínu z lampy) a dvě obsazená jako prázdná (auta na místech jsou malá a parkují hodně vepředu, čímž je ještě zmenšena už tak malá prohledávací plocha).

Tabulka 3: Statistika detekce parkovacích míst snímku 3

řada	Počet míst	Počet obsazených míst	Počet detekovaných míst	Počet chybně detekovaných míst	Procentuální úspěšnost (%)	Číslo obrázku
1	15	9	8	3	80	37
2	26	11	11	0	100	38
3	15	8	8	0	100	39



Obrázek 37: Parkoviště č. 3, řada první



Obrázek 38: Parkoviště č. 3, řada druhá

Obrázek 39: Parkoviště č. 3, řada třetí

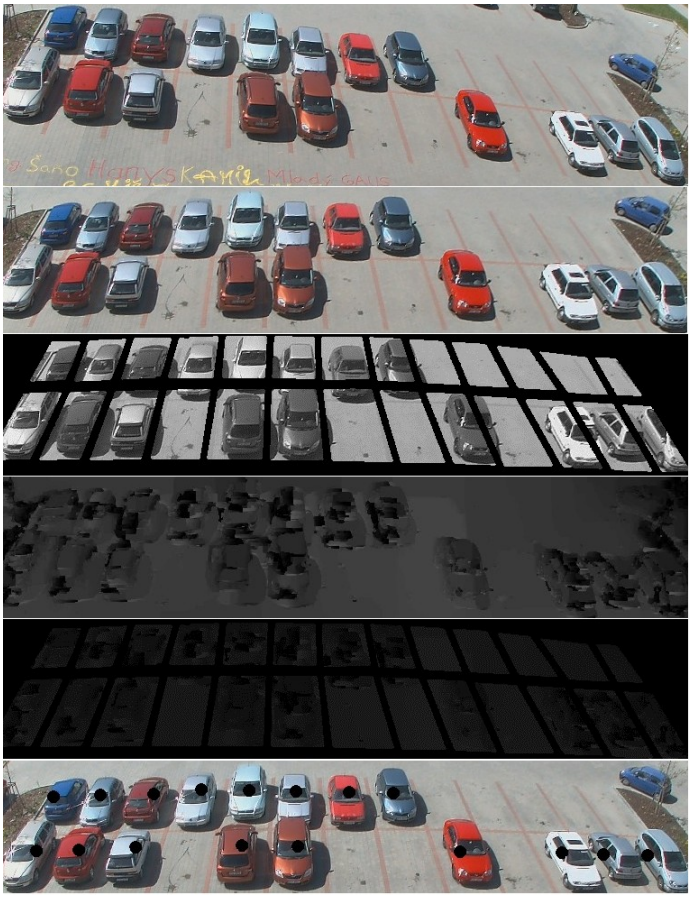
4.8.1.4 ***Snímek parkoviště číslo 4***

Snímek pořízen za slunečného počasí, obsahuje stíny. Ve druhé řadě je obsažen velký flek, který by mohl být detekován jako obsazené parkovací místo. Chybně je detekováno jedno zaplněné místo, které je označeno jako volné.

Tabulka 4: Statistika detekce parkovacích míst snímku 4

řada	Počet míst	Počet obsazených míst	Počet detekovaných míst	Počet chybně detekovaných míst	Procentuální úspěšnost (%)	Číslo obrázku
1	15	11	10	1	93,3	40
2	26	17	17	0	100	41
3	15	10	10	0	100	42

Obrázek 40: Parkoviště č. 4, řada první



Obrázek 41: Parkoviště č. 4, řada druhá

Obrázek 42: Parkoviště č. 4, řada třetí

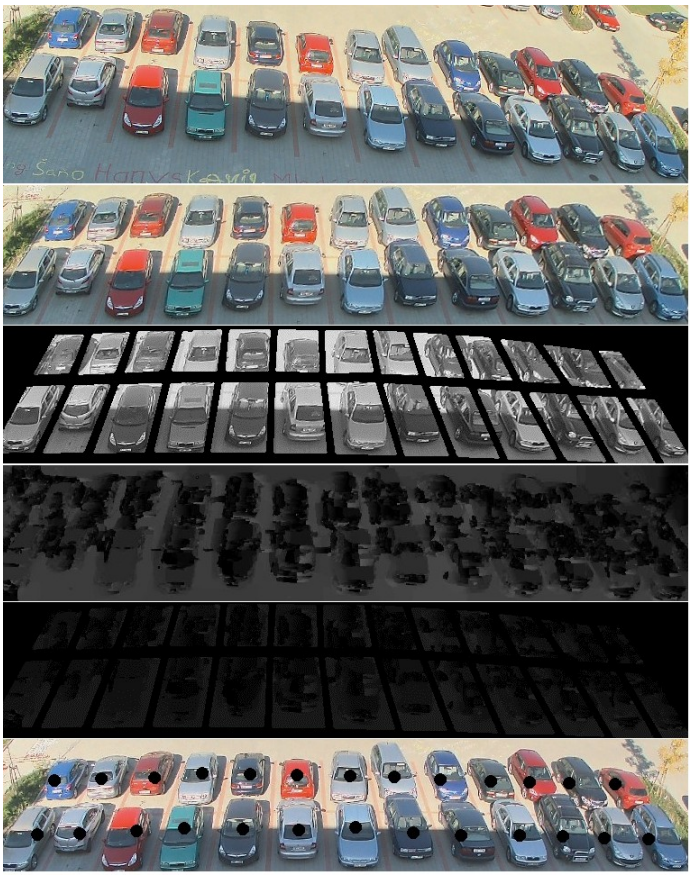
4.8.1.5 ***Snímek parkoviště číslo 5***

Snímek je z poloviny zakrytý stínem, parkoviště je plně obsazené a detekce správná.

Tabulka 5: Statistika detekce parkovacích míst snímku 5

řada	Počet míst	Počet obsazených míst	Počet detekovaných míst	Počet chybně detekovaných míst	Procentuální úspěšnost (%)	Číslo obrázku
1	15	15	15	0	100	43
2	26	26	26	0	100	44
3	15	15	15	0	100	45

Obrázek 43: Parkoviště č. 5, řada první



Obrázek 44: Parkoviště č. 5, řada druhá

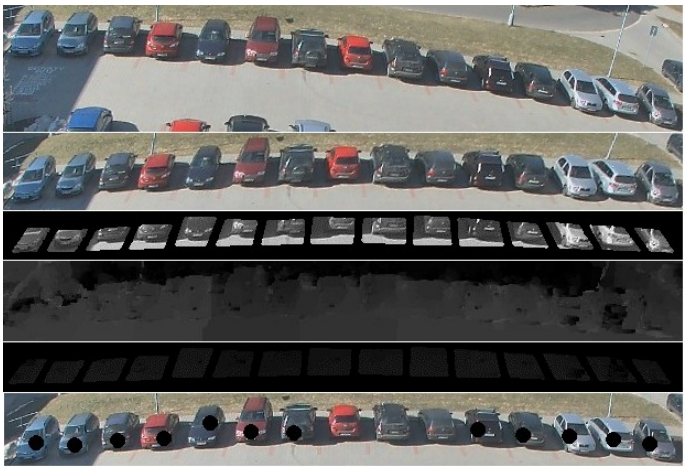
Obrázek 45: Parkoviště č. 5, řada třetí

4.8.1.6 ***Snímek parkoviště číslo 6***

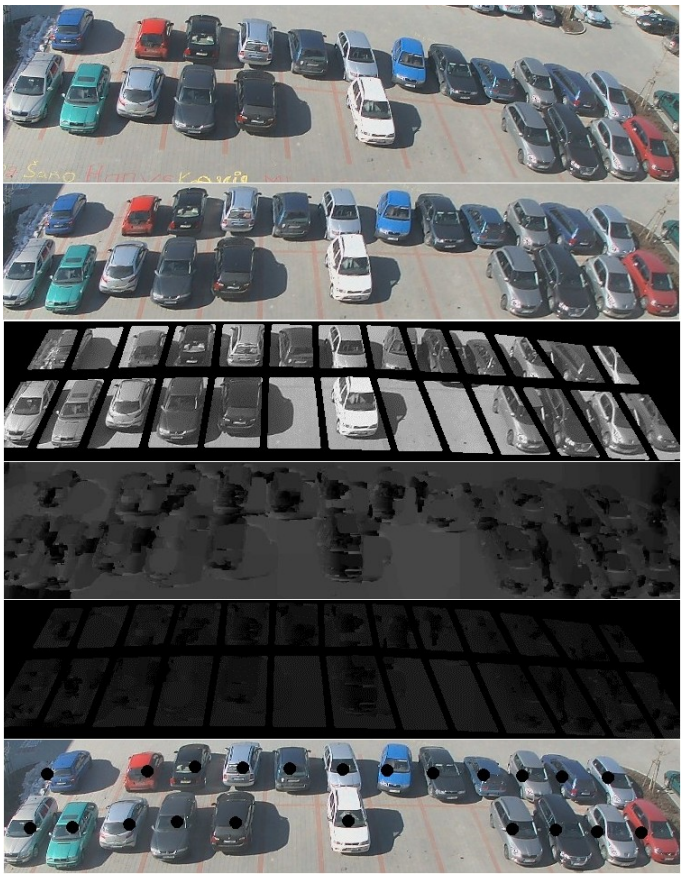
Snímek je pořízen v zimním období, ve třetí řadě je v pravém i levém krajním parkovacím místě nahrnuto hodně sněhu a jsou označena jako obsazená, toto nepovažuji za chybu. Snímek obsahuje hodně stínu. V tomto snímků jsou čtyři chybné detekce. Tři z nich v první řadě, kdy se v jednom případě jedná o malé auto a v případě druhém a třetím se jedná o tmavá auta, která jsou vyhodnocena jako stín. Všechna tři zabraná místa jsou vyhodnocena jako volná. Čtvrtá chyba se nachází ve třetí řadě, kdy je vlivem problematické pozice u levého kraje a sněhu vyhodnoceno volné místo jako zaplněné.

Tabulka 6: Statistika detekce parkovacích míst snímku 6

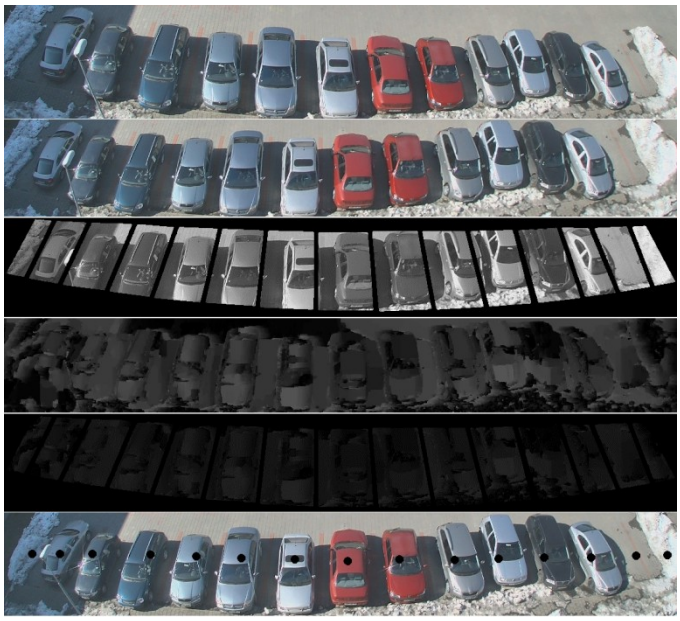
řada	Počet míst	Počet obsazených míst	Počet detekovaných míst	Počet chybně detekovaných míst	Procentuální úspěšnost (%)	Číslo obrázku
1	15	15	12	3	80	46
2	26	22	22	0	100	47
3	15	14	15	1	93,3	48



Obrázek 46: Parkoviště č. 6, řada první



Obrázek 47: Parkoviště č. 6, řada druhá



Obrázek 48: Parkoviště č. 6, řada druhá

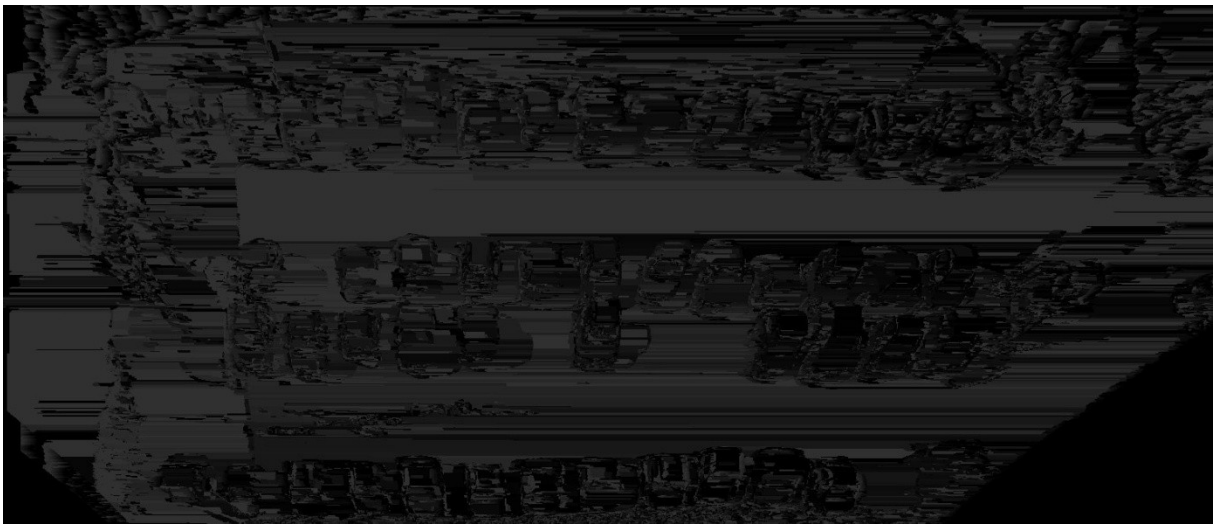
4.8.2 Časová náročnost

Řešení grafových řezů je časově náročné a doba zpracování závisí na velikosti snímků, na kterých je prováděn stereo matching a počtu disparit. Pokud zkombinujeme velký snímek s velkým počtem disparit, budeme čekat na výsledek velmi dlouho a možná nastanou i problémy s alokací paměti. Je třeba si uvědomit, že každému pixelu obrazu je nutné vytvořit stejný počet uzlů grafů jako je disparit a nad tímto grafem pak provádět řez.

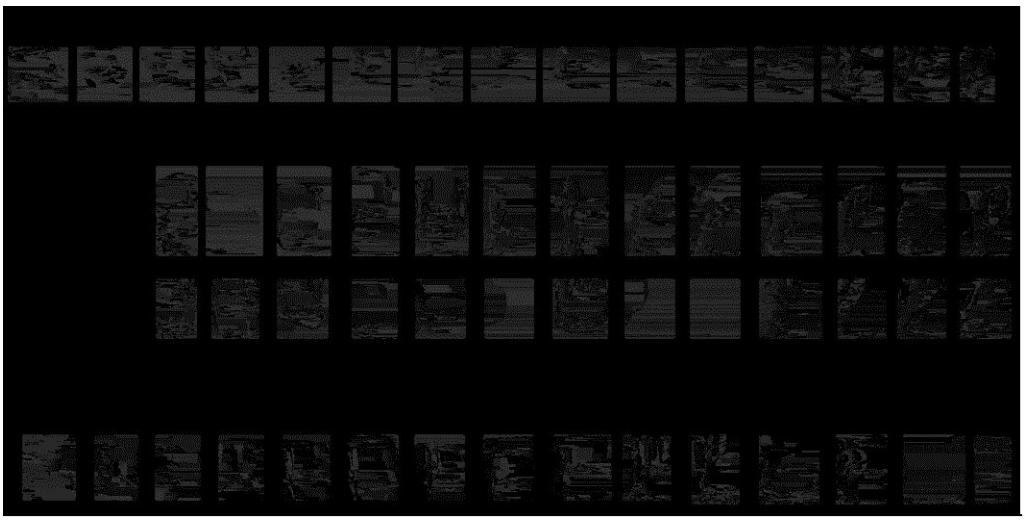
Já jsem pracoval se snímky velikosti 1152x648 a množinou disparit čítající 37 prvků, tvorba mapy disparit trvala přibližně 256 sekund. Při použití odstranění distorze došlo ke zvětšení snímku na 1852x648 a čas výpočtu mapy disparit se prodloužil o 120 sekund. Při pokusech o vyhledávání korespondujících pixelů ve více úrovních byla časová i paměťová náročnost obrovská.

4.9 Alternativní řešení s redukovaným využitím grafových řezů

S plným využitím grafových řezů jsem nedosáhl až tak uspokojivých výsledků, jaké jsem očekával. Rozpoznávání volných parkovacích míst nebylo stoprocentně úspěšné a vyskytovalo se i více chyb, navíc doba časového výpočtu byla pro praktičtější využití moc vysoká. Nejvíce času zabírá optimalizační krok pomocí *α-expanze* při použití náročnějšího výpočtu *smoothCost* termu. Pokud tento term redukuji a nebudu řešit hladkost výsledného obrazu, dojde k výraznému urychlení výpočtu. Částečná hladkost je dodána až po výpočtu řezu. Také jsem v tomto případě použil mnohem „násilnější“ rektifikaci, kdy došlo k velké deformaci snímků a potlačení jejich stereo vlastností. Výsledkem byly snímky, které opticky vypadaly jako snímané z prostoru přímo nad parkovištěm a odstranil jsem i distorzi. Výsledná mapa disparit byla sice opticky nepřehledná (obrázek 49), ale detekce byla úspěšnější (obrázek 51) a s menší časovou náročností.



Obrázek 49: Mapa disparit alternativním způsobem



Obrázek 50: Alternativní mapa disparit v parkovacích místech

Obrázek 51: Detekce obsazenosti alternativním způsobem

5 Závěr

Cílem této diplomové práce bylo prozkoumat využití grafových řezů pro detekci volných parkovacích míst ze stereo obrazů. Pro tento úkol bylo potřeba získat vědomosti o možnostech a způsobech úpravy obrazu, dosavadním výzkumu a využití grafových řezů v počítačovém vidění a získat přehled o knihovnách pro zpracování obrazu a grafových řezů a naučit se s nimi pracovat. Následně aplikovat získané vědomosti a naprogramovat aplikaci pro detekci obsazenosti parkoviště ze stereo obrazů využitím grafových řezů.

5.1 Přínos práce

Po prozkoumání a praktickém vyzkoušení této metody na detekci obsazenosti parkoviště jsem došel k závěru, že tento postup není k tomuto účelu příliš vhodný. Pro větší úspěšnost rozpoznání stavu parkovacích míst by bylo potřeba technicky složité a nákladné řešení pořizování stereo snímků tak, aby byly snímky pro tuto metodu vhodné. Další nepraktická vlastnost je poměrně velká časová náročnost zpracování a vytvoření mapy disparit parkoviště, kdy zpracování trvá minuty. Vzhledem snímkům parkoviště, se kterými jsem pracoval, bylo potřeba zvolit vhodně mnoho konstant, kdy i malá změna vedla k odlišným konečným výsledkům, což je také nepraktické.

5.2 Chyby a vylepšení

Oblast, na které by určitě bylo možné zapracovat je rychlost výpočtu a vytvoření mapy disparit použitím paralelizace, kdy se snímek rozloží na části například po jednotlivých parkovacích řadách a mapa disparit se pro každou řadu bude počítat separátně, čímž dojde k velkému urychlení. Další vylepšení se týká vstupů, kdy by stereo obrazy byly pořizovány způsobem vhodným pro použití této metody, čímž by došlo k zlepšení úspěšnosti detekce.

5.3 Shrnutí

Vyzkoušel jsem detekci obsazenosti parkoviště pomocí metody grafových řezů, sepsal své poznatky k výhodám i nedostatkům a okomentoval dosažené výsledky.

Detekce volných parkovacích míst a její případné budoucí propojení s parkovacími systémy aut má jistě smysl a budoucnost. Integrace detekčního systému například do bezpečnostního kamerového systému by bylo velkou výhodou, proto je potřeba pokračovat v dalším zkoumání.

Použitá literatura

[1] ZUREIKI, Ayman, Michel DEVY a Raja CHATILA. Stereo Matching and Graph Cuts, Stereo Vision. Stereo Vision. InTech, 2008-11-01. DOI: 10.5772/89. Dostupné z: http://www.intechopen.com/books/stereo_vision/stereo_matching_and_graph_cuts

[2] ROY, S. a I. J. COX. A maximum-flow formulation of the N-camera stereo correspondence problem. Sixth International Conference on Computer Vision (IEEE Cat. No.98CH36271). Narosa Publishing House, 1998, s. 492-499. DOI: 10.1109/ICCV.1998.710763. Dostupné z: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=710763

[3] VEKSLER, O. Efficient graph-based energy minimization methods in computer vision. PhD Thesis, Cornell University. Adviser : Ramin Zabih. 1999.

[4] KOLMOGOROV, V. a R. ZABIH. Computing visual correspondence with occlusions using graph cuts. *Proceedings Eighth IEEE International Conference on Computer Vision. ICCV 2001*. IEEE Comput. Soc, 2001, s. 508-515. DOI: 10.1109/ICCV.2001.937668. Dostupné z: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=937668

[5] MAREŠ, Martin. *Krajinou grafových algoritmů: průvodce pro středně pokročilé*. Vyd. 1. Praha: ITI, 2007. 71 s. ISBN 978-80-239-9049-2. Dostupné z: www.mff.cz/data/ADS2-Mares_KGA.pdf

[6] GOLDBERG, A. V. & TARJAN, R. E. (1988). A new approach to the maximum-flow problem. Journal of the Association for Computing Machinery (JACM), 35(4):921-940.

[7] KOLMOGOROV, V. & ZABIH, R. Efficient Approximate Energy Minimization via Graph Cuts. Y. Boykov, O. Veksler, R.Zabih. IEEE TPAMI, 20(12):1222-1239, Nov 2001.

[8] BOYKOV, Y. a V. KOLMOGOROV. An experimental comparison of min-cut/max- flow algorithms for energy minimization in vision.*IEEE* Transactions *on Pattern Analysis and Machine Intelligence*. 2004, vol. 26, issue 9, s. 1124-1137. DOI: 10.1109/TPAMI.2004.60. Dostupné z: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1316848

[9] ČERNÝ, Jakub. Základní grafové algoritmy: Toky v sítích. 2010, 143 - 180. Dostupné z: http://kam.mff.cuni.cz/~kuba/ka/ka.pdf

[10] MANJUNATH, B. S.Dostupné z: www.ece.ucsb.edu/~manj/ece181bS04/L14(morestereo).pdf

[11] LOMBAERT, Herve. Energy minimization with Graph Cuts. [online]. 2006 [cit. 2013-07-29]. Dostupné z: http://step.polymtl.ca/~rv101/energy/

[12] ŘÍHA, Kamil a Petr HUJKA. *Epipolární geometrie* [online]. 2005 [cit. 2013-07-30]. Dostupné z: http://www.elektrorevue.cz/clanky/05017/index.html#homografie

[13] BENEDA, Martin. *Homografie a epipolární geometrie* [online]. 2010 [cit. 2013-07-30]. Dostupné z: http://trilobit.fai.utb.cz/homografie-a-epipolarni-geometrie

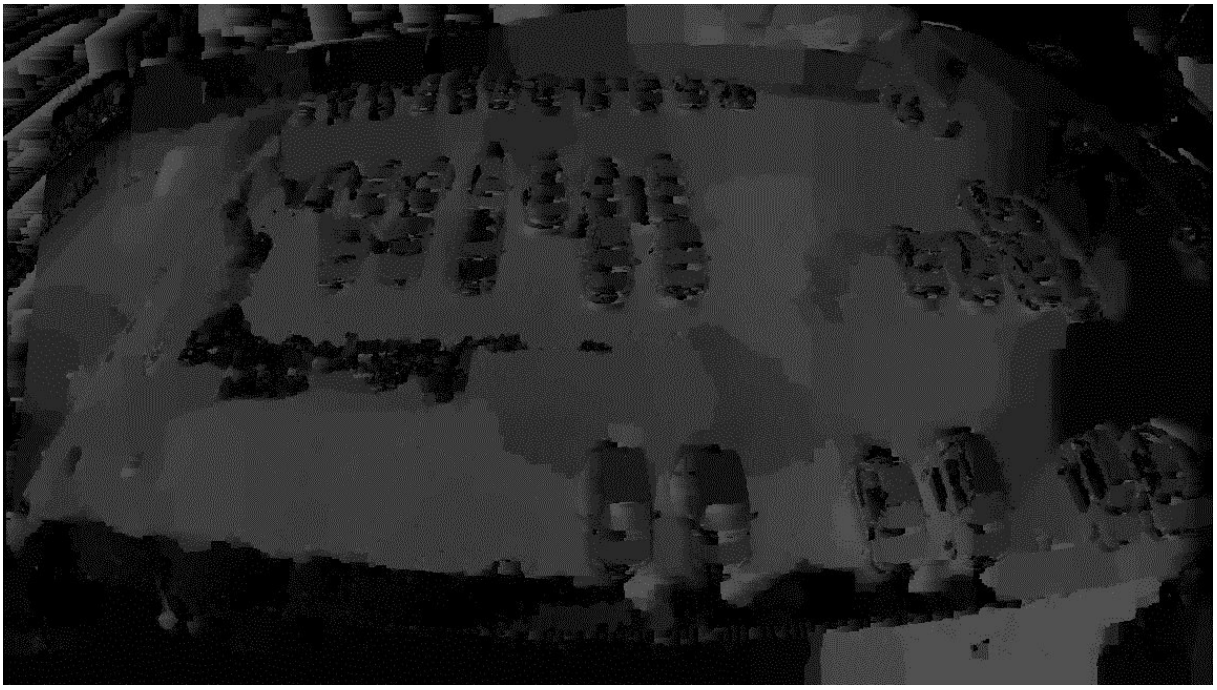
Seznam příloh

Součástí DP je CD.

Adresářová struktura přiloženého CD:

- code_files
  - source_files
  - header_files
  - resource_files
- disparity_map_examples
- parking_lot_images
- other_images
- texts

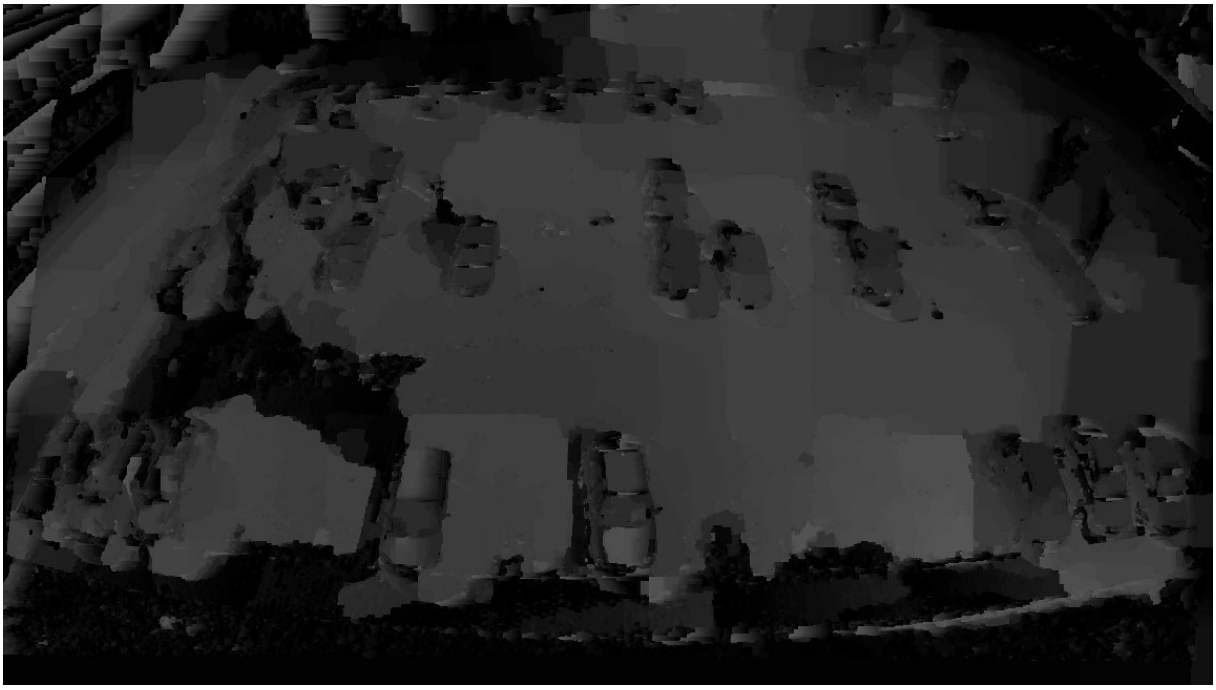
Příloha.A: Snímky parkoviště1



Příloha.B: Snímky parkoviště 2

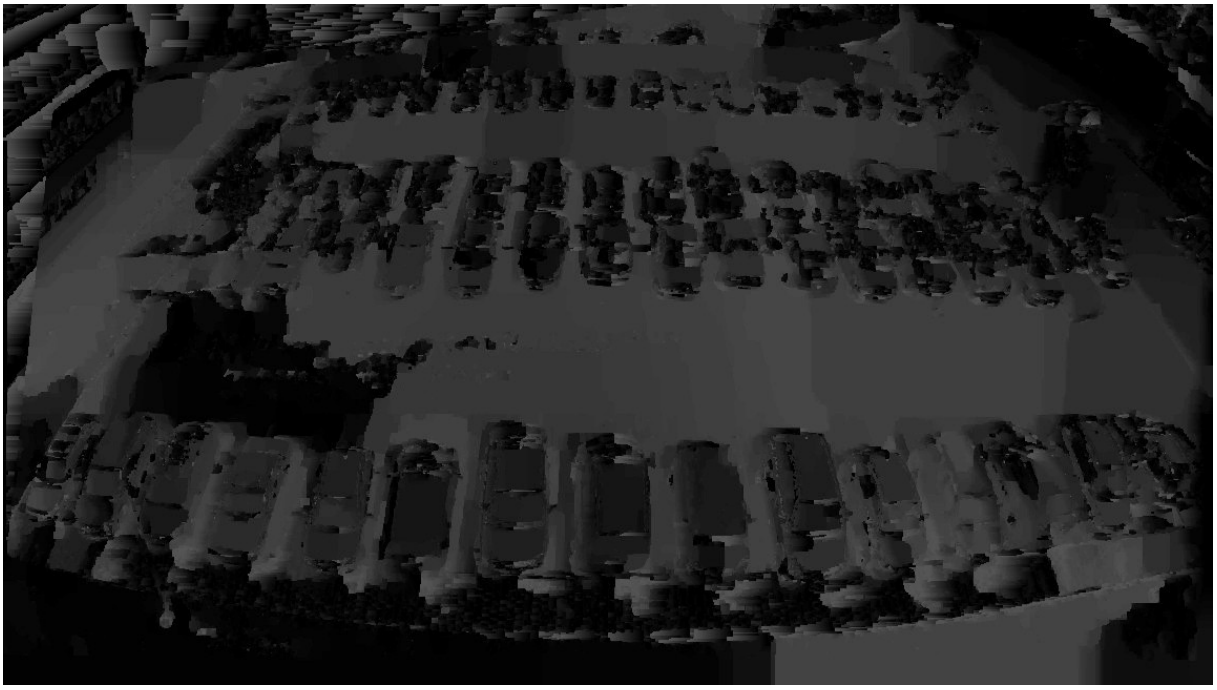

Felipe

Příloha.C: Snímky parkoviště 3



Příloha.D: Snímky parkoviště 4

Příloha.E: Snímky parkoviště 5



Příloha.F: Snímky parkoviště 6

Příloha.G: Nastavení parametrů aplikace

Parametry konzolové aplikace jsou uloženy v souboru settings.txt, tyto parametry je možné měnit a zkoušet tak dosáhnout co nejlepších výsledků v detekci obsazenosti. Krom cest k načítaným souborům a cest k ukládání vytvořených snímků je možně nastavovat parametry pro vyhledávání disparit v jednotlivých částech snímku a odhad šumu v jednotlivých řadách snímku. Očekávané rozměry snímků jsou 1152x648.